Proyecto Fin de Grado

Herramienta de autoría de ayuda al diseño conceptual de videojuegos educativos

Grado en Ingeniería Informática

Curso 2013/2014

Autor: Alfredo Alba Mansilla

Tutor: Telmo Zarraonandia Ayo

AGRADECIMIENTOS

Quiero dedicar este Trabajo Fin de Grado a una serie de personas que me han apoyado y ayudado a lo largo de todo este tiempo:

A mis padres, porque sin su apoyo y su esfuerzo durante toda mi vida no habría sido posible llegar hasta donde he llegado hoy. Espero que os sintáis orgullosos de vuestro trabajo.

A mis hermanas, ya que han estado siempre ahí para todo lo que he necesitado y siempre se han preocupado por su hermano pequeño. No se puede tener hermanas mejores.

A Natalia, mi mejor amiga y compañera. Gracias por todos los buenos ratos que me has brindado durante estos años de carrera y por todo lo que hemos aprendido juntos. Espero que sea así siempre.

A Dani, porque te cruzaste en mi camino sin avisar. Eres la persona que más me ha tenido que aguantar, tanto en los momentos buenos como en los malos y siempre has estado ahí. Gracias por escucharme siempre y distraerme cuando más lo necesitaba. A estas alturas creo que sabes del proyecto tanto como yo.

A Telmo, por demostrar su interés en mi trabajo desde el primer día y ayudarme en todo lo que necesitaba. Espero que el resultado haya sido de tu agrado.

En última instancia, a todas las personas que para bien o para mal he conocido durante estos años, ya que entre todos habéis contribuido a que sea tal y como soy.

RESUMEN

La educación es un tema de gran importancia entre docentes y padres. En un momento en que la tecnología forma parte de la vida cotidiana, se intenta aprovechar sus beneficios en el ámbito escolar, como es el caso de los videojuegos.

Los videojuegos han adquirido en los últimos años una gran importancia dentro de la industria del ocio. Es raro no encontrar una videoconsola en un hogar, sobre todo si en él viven niños o jóvenes. Este interés por los videojuegos ha motivado al personal docente para que intenten incluirlos como parte de su programa educativo, ya que son un complemento útil y ameno para sus alumnos. Sin embargo, crear un videojuego no es sencillo si no se poseen los conocimientos adecuados.

Este proyecto consiste en una herramienta de autoría que ayude al docente a diseñar sus propios videojuegos educativos de una manera sencilla. Los diseños generados con esta herramienta serán utilizados en la plataforma GREP (Game Rules and scEnario Platform) para crear el videojuego educativo.

Palabras clave: Herramienta de Autoría, Videojuego Educativo, Aprendizaje basado en Juegos, Diseño de Videojuegos

ABSTRACT

Education is an extremely important topic among teachers and parents. In a moment in which technology is part of the daily routine, its benefits are tried to take advantage in the school area, such as the video games.

In the last years video games have acquired a great importance inside the entertainment industry. It´s difficult not to find a video game console in a home, especially when children or teenagers live in it. This interest in video games has motived to the teachers in order that they try to include them as a part of his educational program, since they are a useful and pleasant complement for his pupils. However, to create a video game is not simple whether the suitable knowledge is not possessed.

This project consists of an authoring tool that helps the teacher to design his own educational video games easily. The designs generated with this tool will be used in the GREP platform (Game Rules and scEnario) to create the educational video game.

Keywords: Authoring Tool, Educational Game, Game based Learning, Game Design

ÍNDICE

Índice de imágenes

Índice de tablas

1. Introducción

El objetivo de esta sección es proporcionar una visión general del proyecto a tratar en el presente documento. En primer lugar, se presenta el contexto en el que se encuentra el proyecto. A continuación, se muestran los objetivos que se pretenden alcanzar con su consecución y se describen las distintas fases de desarrollo planteadas. Por último, al final del capítulo se presenta la estructura que seguirá el documento.

1.1 Contexto

Un niño dispone de la capacidad innata de aprender desde pequeño y es a través del juego como consigue ponerla en práctica. Desde siempre, padres y educadores ponen a disposición de sus hijos o alumnos todo tipo de herramientas para que puedan ir desarrollando poco a poco sus habilidades a medida que crecen con el objetivo de que sepan desenvolverse en su día a día. Bajo esta premisa surgieron los llamados "juegos educativos".

1.1.1 Juegos educativos

La infinita curiosidad de un niño propicia que explore sin cesar el mundo que le rodea en busca de nuevas experiencias o descubrimientos. Los juegos educativos son una excelente herramienta para ayudar al niño a satisfacer su curiosidad, a la vez que le proporcionan buenas dosis de entretenimiento y de aprendizaje.

Según el portal de recursos de psicología y educación *reEduca.com* "un juego educativo es aquel que tiene un objetivo educativo implícito o explícito para que los niños aprendan algo específico de forma lúdica". De acuerdo con esta definición, ya no es posible entender el juego como una actividad dedicada únicamente al ocio y disfrute de la persona que lo practica.

El juego como tal ha sido estudiado por numerosos investigadores. C. R. Miranda [1] destaca que el juego ayuda al desarrollo social del niño además de contribuir al uso de la imaginación, ya que en grupo los niños son capaces de interactuar entre ellos para representar diferentes escenarios que hayan visto previamente. Por otro lado, asimilan sin darse cuenta roles sociales que pueden practicar con éxito en diferentes momentos de su vida diaria, lo que les motiva a seguir aprendiendo mientras juegan.

C. R. Miranda hace referencia también a la importancia del juego en el desarrollo emocional de los más pequeños [1], ya que al representar situaciones cotidianas consiguen autoconocerse, lo que les servirá para manejar su propia conducta en diferentes momentos. No hay que olvidar tampoco los beneficios que proporciona el juego a la hora de adquirir y mantener habilidades físicas y motoras. Padres y educadores comparten con sus hijos y alumnos experiencias a través del juego, ya sea participando activamente con ellos u observando atentamente su evolución durante el mismo.

Sin embargo, en un mundo en el que la tecnología ha invadido la mayoría de los ámbitos de la vida, el juego tal y como se conocía tradicionalmente ha adoptado nuevas formas de expresión, aunque conserva sus características educativas, sociales e

intelectuales unidas para proporcionar al niño satisfacción y diversión al practicarlo. La forma de expresión más conocida viene dada por los videojuegos.

1.1.2 Los videojuegos y su historia

Los videojuegos constituyen en la actualidad una industria tecnológica muy importante y mueve enormes cantidades de dinero. Sin embargo, esta industria es relativamente joven. Su origen se suele situar en torno a los años 50 [2].

Así, en 1952 nace el primer videojuego de la historia a manos de Alexander S. Douglas, una versión electrónica del popular juego de mesa de "las 3 en raya". En 1966, se comienza a trabajar en el antecesor de Odyssey, la primera videoconsola de sobremesa que fue lanzada en 1972. En ese mismo año se crea *Pong*, uno de los videojuegos más conocidos de la historia y tras su estela aparece *Space Invaders* o la mítica Atari 2600 [2].

Ilustración 1. Atari 2600

Los 80´s son conocidos como la "década de los 8 bits". Se produce un estallido de lanzamientos de videoconsolas, como la Sega Master System o la NES. No obstante, son los ordenadores de 8 bits los que acaparan la atención en aquellos años, destacando el *Sinclair ZX Spectrum*, el *Comodore* o el *Amstrad*. La aparición en 1985 de *Super Mario Bros* marca un antes y un después en una generación enamorada de títulos como *Pacman* [2].

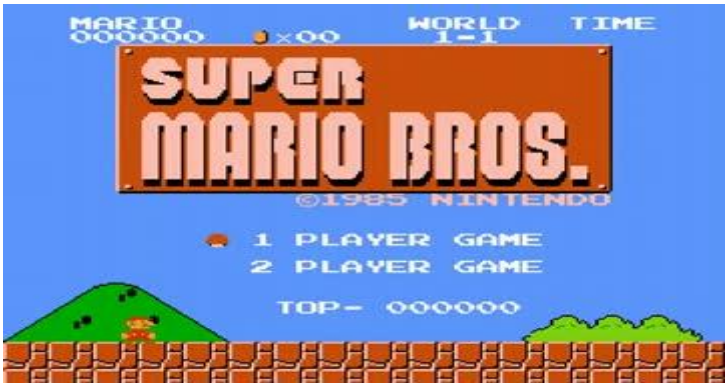


Ilustración 2. Super Mario Bros 1985

En los 90 se produce el salto a los 16 bits que, junto al nacimiento del CD-ROM, propician la creación de los primeros juegos 3D utilizando el PC como plataforma hardware. Estas tecnologías son rápidamente aprovechadas por compañías como Nintendo, Sony o Sega. Las videoconsolas portátiles y de sobremesa, junto con las posibilidades que abrió Internet para el PC como plataforma, desplazan a las recreativas del mercado [2].

Con el nuevo siglo, la industria de los videojuegos se establece como tal. En el año 2000, Sony lanza la Play-Station 2. Un año después, Microsoft se une a la aventura creando la XBox y Nintendo lanza al mercado Game Cube y Game Boy Advance [2].

En 2003, Nokia entra en el mercado de las consolas portátiles con N-Gage, un híbrido de móvil y consola que no cosechó mucho éxito. En 2004, Nintendo comienza a distribuir la Nintendo DS (Dual Screen), que apostaba por la jugabilidad y la interacción con el usuario a través de una pantalla táctil, y Sony lanza la PSP (Play-Station Portable), con prestaciones similares a la PSX.

Las tres importantes compañías continúan compitiendo años más tarde con Xbox 360 (Microsoft), Play-Station 3 y PSP Vita (Sony) y Nintendo 3DS y Wii (Nintendo). La nueva y actual generación de videojuegos comienza en 2012 cuando Nintendo distribuye Wii U, la sucesora de Wii, responsable de crear nuevos paradigmas de juego en familia. En 2013 aparecen en el mercado casi simultáneamente XBox One y Play-Station 4, las videoconsolas de nueva generación de Microsoft y Sony, respectivamente.

Ilustración 3. Wii, PS3 y XBox 360

A día de hoy, los videojuegos no sólo se comercializan para videoconsolas, sino que pueden ser encontrados en multitud de plataformas, como ordenadores, tablets, smartphones, etc., por lo que es más sencillo su uso para fines educativos. La idea de una herramienta que permita a docentes crear sus propios videojuegos educativos no es descabellada teniendo en cuenta las posibilidades que se ofrecen.

1.2 Objetivo

Una vez presentado el contexto, es momento de explicar el objetivo del proyecto, el cual consiste en el desarrollo de una herramienta de autoría que sirva de apoyo al diseño conceptual de videojuegos educativos.

El auge de la tecnología, y en especial, de los videojuegos, en la sociedad actual ha propiciado el comienzo de la modernización de los paradigmas de educación clásicos. Algunos docentes han empezado a crear sencillos videojuegos educativos con los que conseguir que sus lecciones sean más amenas, pero sin olvidar que la meta final es el aprendizaje de las materias por parte de sus alumnos.

No obstante, de todo esto surge un problema relevante: la mayoría del personal docente de las instituciones educativas no dispone de los conocimientos técnicos necesarios para diseñar e implementar con éxito sus propios videojuegos, lo que dificulta el avance de esta novedosa tendencia y favorece la impartición de las clases del modo habitual.

El proyecto nace con la intención de remediar este problema y acercar a los educadores una herramienta que minimice la asistencia técnica durante el proceso de diseño del videojuego y que sea sencilla de utilizar.

Para ello, se necesitan unas pautas que ayuden al docente a diseñar su videojuego, las cuales se van a basar en el modelo conceptual de videojuegos educativos basado en reglas y escenarios (GREM). También se necesita una plataforma que interprete el diseño del docente y genere el entorno virtual en el que el alumno pueda jugar. Esta plataforma es la denominada GREP (Game Rules and scEnario Platform).

1.2.1 Modelo GREM

El modelo GREM (Game Rules and scEnario Model) [3] fue propuesto por Zarraonandia, Díaz, Aedo y Ruíz en 2012. Este modelo surge a partir de la necesidad de definir de una manera clara el conjunto de componentes fundamentales de un videojuego educativo (EG, Educational Game). Para facilitar la reutilización de diseños previos de videojuegos educativos y facilitar y acelerar la creación de nuevos diseños el modelo considera dos perspectivas de diseño distintas: reglas de juego y escenario de juego.

Las reglas de juego describen la mecánica del videojuego, es decir, cómo ha de ser jugado. El escenario constituye el medio virtual en que tendrá lugar la acción del videojuego, la interfaz necesaria para interactuar con él y un conjunto de servicios disponibles. Estos dos ámbitos son independientes entre sí, lo que puede dar lugar a que un mismo conjunto de reglas pueda ser usado en diferentes escenarios y un mismo escenario en diferentes juegos.

Cada ámbito constituye un submodelo dentro del modelo GREM y cada uno se compone de diferentes elementos, como puede observarse en la imagen mostrada en la siguiente página.

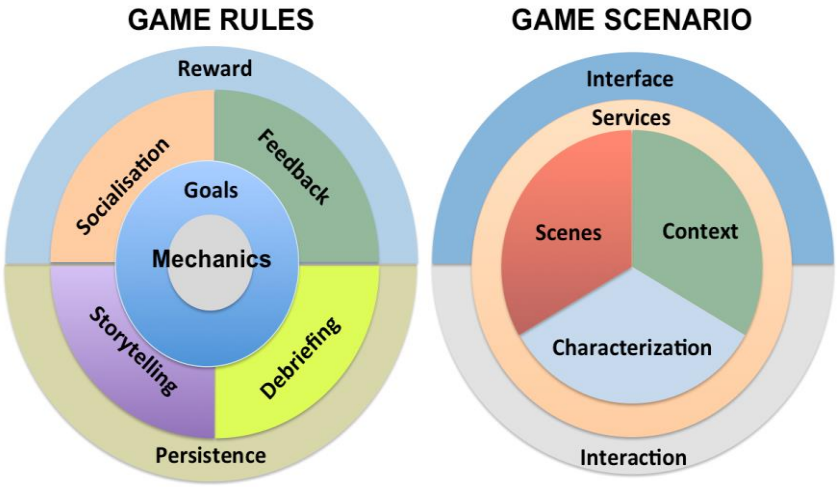


Ilustración 4. Elementos del modelo de reglas y del modelo de escenarios

Modelo de reglas de juego:

- Nivel 1. Mecanismos. Son las entidades que pueden aparecer en el juego, su estado y las acciones que pueden ejecutar cada una de ellas.

- Nivel 2. Objetivos. Constituyen la meta que se pretende alcanzar con el videojuego y cómo ésta ha de alcanzarse.

- Nivel 3. Feedback, socialización, storytelling y debriefing. Cada uno de los términos tiene un significado distinto dentro del videojuego.

    - Feedback. Información que se mostrará cuando se cumplan ciertas condiciones en el videojuego y que puede tener una finalidad motivacional o educativa.
    - Socialización. Define la interacción social que se produce en el videojuego.
    - Storytelling. Se compone de un conjunto de episodios que se suceden en un determinado orden dentro del videojuego y que indican el conjunto de objetivos que ha de alcanzar el jugador en cada uno.
    - Debriefing. Se utiliza para conectar las lecciones aprendidas dentro del videojuego con su aplicación en el mundo real.

- Nivel 4. Recompensa y persistencia.

    - Recompensa. Como su propio nombre indica, representa las recompensas que el jugador puede obtener al superar ciertos objetivos, como episodios especiales.
    - Persistencia. Define el alcance de cada episodio y puntos de guardado y restauración en cada uno de ellos. Con ello, se posibilita que el jugador pueda conservar los progresos realizados en el juego y así retomarlo en otro momento.

Modelo de escenarios de juego:

- Nivel 1. Representaciones del juego. Pueden dividirse en tres elementos:
  - Escena. Provee un medio físico o situación en que la acción del juego puede tener lugar a través de entidades de escenario con las que el jugador puede interactuar, como cofres del tesoro, elementos de fondo como estalagmitas o enlaces entre cuevas como puentes de madera.
  - Personajes. Entidades manejadas por el jugador.
  - Elementos de contexto, como audio, animación, etc.

- Nivel 2. Servicios. Conjunto de funcionalidades que proporcionan algún servicio de utilidad al jugador, como chats, foros, tablas de mensajes, etc.

- Nivel 3. Interfaz de juego e interacción.
  - Interfaz de juego, compuesta de elementos como ventanas, tabs, check boxes, etc.
  - Interacción. Define modos de interactuar del jugador con el videojuego. La interacción puede ser física si requiere el uso de dispositivos físicos, como las teclas del ordenador, o virtual si está relacionada con la interfaz de juego.

1.2.2 Plataforma GREP

La plataforma GREP (Game Rules and scEnario Platform) interpreta diseños de videojuegos educativos basados en el modelo GREM. Estos diseños se encuentran especificados en ficheros XML.

Esta plataforma ha sido desarrollada mediante el motor de videojuegos Unity 3D como parte de otros trabajos fin de grado: “Desarrollo de un juego educativo de lógica configurable”, de Daniel Barba Castaño, y “Herramienta de definición de comportamientos de personajes no jugadores en videojuegos educativos”, de Javier Abellán Fernández.

A partir del diseño, GREP genera el correspondiente entorno virtual que permite al alumno jugar al juego y almacena información sobre la partida para poder conocer el desarrollo de la misma.

Las imágenes que se presentan a continuación se corresponden a un videojuego educativo generado directamente con la plataforma GREP:

Ilustración 5. Videojuego generado con la plataforma GREP

Ilustración 6. Videojuego generado con la plataforma GREP (2)

Ilustración 7. Videojuego generado con la plataforma GREP (3)

1.2.3 Objetivos específicos del proyecto

Este proyecto tiene por objetivo proporcionar al educador una herramienta de autoría que guiará al usuario a través de la definición de cada uno de los niveles de diseño conceptual de un videojuego educativo del modelo GREM. Las funcionalidades principales de la herramienta serán:

- **Creación de conjuntos de reglas**. La herramienta ofrecerá la posibilidad de recorrer cada uno de los niveles del modelo de reglas de juego para crear todos los elementos del diseño del videojuego.

- **Creación de escenarios**. Del mismo modo que con las reglas, la herramienta permitirá la producción de escenarios pasando por todos los niveles detallados en el modelo de escenarios de juego.

- **Unión de reglas y escenarios**. La herramienta propiciará la unión de reglas y escenarios creados previamente para dar lugar al diseño final del videojuego.

La herramienta permitirá exportar el diseño final producido en forma de un conjunto de documentos XML (eXtensive Markup Language) [4]. Estos ficheros podrán ser luego interpretados por la plataforma GREP (Game Rules scEnario Platform), desarrollada mediante el motor de videojuegos Unity 3D [5] y que genera un entorno virtual 3D en el que el juego pueda ser jugado por el usuario.

1.3 Estructura del documento

El documento se encuentra estructurado en varios capítulos, tal y como se muestra a continuación:

- **Introducción**. Constituye la presentación del proyecto en la que se da una visión global acerca de él. En ella se muestra el contexto que propicia su desarrollo, su objetivo principal y las fases de desarrollo que se han manifestado. Termina la sección con la descripción de la estructura del documento.

- **Estado del arte**. En esta sección se plantean los principales problemas que surgen alrededor de la unión de videojuegos y educación, se muestran algunas herramientas de autoría y se realiza un análisis de tecnologías para elegir las más acordes al desarrollo del proyecto.

- **Gestión del proyecto**. En esta sección se presenta el ciclo de vida que se ha utilizado en el proyecto, la planificación inicial y los costes derivados del desarrollo.

- **Análisis**. En esta sección se realiza una descripción del sistema y se incluyen los requisitos de usuario y de software identificados, los casos de uso pertinentes y las matrices de trazabilidad derivadas.

- **Diseño**. En esta sección se detalla la arquitectura escogida, los componentes que conforman el sistema y un diseño de la interfaz a bajo nivel que ilustra el aspecto final de la herramienta.

- **Implementación**. Sirve para mostrar los aspectos clave de la programación de la herramienta.

- **Pruebas**. Recoge las baterías de pruebas necesarias para verificar el funcionamiento adecuado de la herramienta.

- **Planificación final y desviaciones sobre la planificación inicial**. Indica los posibles percances acontecidos a lo largo del desarrollo del proyecto que hayan impedido respetar la planificación inicial al pie de la letra.

- **Conclusiones**. En esta sección se localizan las conclusiones personales derivadas del desarrollo del proyecto, las futuras líneas de investigación y posibles mejoras de la herramienta.

- **Referencias bibliográficas**. Reúne todas las fuentes de información que se han utilizado para dar consistencia al texto escrito en el presente documento.

- **Glosario de términos**. Recopila y define términos cuyo significado no pueda ser comprendido de primera mano por el lector del documento.

2. Estado del arte

2.1 Introducción

Esta sección tiene como objetivos plantear el problema que ha dado lugar a la elaboración del presente proyecto y los trabajos que se han llevado a cabo previamente para tratar de encontrarle una solución. También se hará un análisis de tecnologías útiles para la implementación del prototipo central del proyecto y, finalmente, se elegirán aquellas más acordes con sus requerimientos.

El prototipo mencionado consiste en una aplicación para *tablets* [6] que sirva de soporte a las tareas de diseño conceptual de un videojuego. En otras palabras, se trata de una herramienta de autoría destinada a facilitar la creación de videojuegos educativos.

2.2 Planteamiento del problema

Actualmente, la sociedad no concibe su vida alejada de las tecnologías de la información y la comunicación (TIC). La creación de los primeros ordenadores y, años más tarde, la aparición de Internet, han propiciado un afán incontrolable por acceder a la información y mantenerse al día de lo que sucede en todas partes del mundo. Es más, tal es el volumen de información existente en la red que el usuario puede llegar a sentirse desbordado al acceder a ella.

Con el surgimiento de los ordenadores, el ocio electrónico vio la luz y los videojuegos se instalaron en la vida cotidiana del ser humano, sobre todo entre los más jóvenes. Tan alta es la variedad de videojuegos disponibles que los adolescentes (y no tan adolescentes) siempre encontrarán un título acorde a sus gustos al que dedicarán tiempo de su día a día.

Videojuegos aparte, es momento de hablar de la educación. En los últimos años, el sistema educativo ha ido incorporando las tecnologías de la información en sus aulas y se han facilitado herramientas que permiten, por ejemplo, el intercambio de información entre docentes y alumnos a través de la red como Moodle [7], incorporado en muchos campus virtuales universitarios.

Los videojuegos y la educación guardan una estrecha relación hoy en día. B. G. Salvat subraya la importancia de los videojuegos como material educativo [8] y los considera un medio de acceso a la cultura por parte de los niños. Los juegos electrónicos permiten a los alumnos adquirir diferentes tipos de habilidades, fomentan las relaciones de compañerismo entre ellos, facilitan el aprendizaje e incitan a la reflexión.

No obstante, el proceso de creación de un videojuego no es simple porque requiere de ciertos conocimientos técnicos, como un nivel de programación avanzado. La mayoría del personal docente de las escuelas está en contacto directo con las TIC, pero no dispone de los conocimientos técnicos necesarios para crear sus propios juegos electrónicos, por lo que es difícil la inclusión de estos en sus clases.

Llegados a este punto, es preciso plantearse la pregunta que representa el problema: "¿Cómo conseguir que los educadores puedan diseñar y desarrollar sus propios videojuegos?".

Las herramientas de autoría surgen con la intención de paliar este problema. Una herramienta de autoría es una aplicación informática que posibilita la creación y gestión de materiales educativos en formato digital. Estas herramientas pueden ser utilizadas por el personal docente para desarrollar sus juegos sin recurrir a conocimientos técnicos concretos.

Por lo tanto, la intención de este proyecto es presentar una herramienta de autoría que permita diseñar videojuegos de una manera intuitiva y sencilla.

2.2.1 Binomio videojuego-educación

Los videojuegos se han convertido en un elemento indispensable de ocio en los hogares. Niños, jóvenes y adultos dedican asiduamente tiempo a disfrutar de las posibilidades que ofrecen cada vez que necesitan aislarse de la rutina diaria.

Muchos padres suelen imponer restricciones de tiempo a sus hijos para evitar que los videojuegos generen en ellos una cierta dependencia que les pueda apartar de sus quehaceres diarios. R. Bustillo se pregunta por qué los videojuegos atraen tanto a los jóvenes [9] y llega a la conclusión de que este hecho tiene origen en el reto que suponen para ellos. Si a este reto se le añaden ciertos elementos que en la vida real el niño o joven no puede ejecutar, como magias o fuerza sobrehumana, e incluso adquirir el rol de un personaje carismático y admirable, se creará el caldo de cultivo perfecto para mantenerlo durante horas frente al ordenador o la televisión.

Como se ha mencionado antes, los videojuegos fueron pensados inicialmente como una alternativa de ocio, por lo que no es de extrañar que el niño encuentre en ellos un modo de desconexión de su vida diaria repleta de actividades que pueden no ser de su agrado.

Sin embargo, los videojuegos no sólo sirven para entretener. Se ha demostrado que proporcionan una serie de beneficios que pueden contribuir al desarrollo del niño. Por un lado, favorecen aspectos como la memorización, la percepción espacial y el razonamiento. Por otro lado, las habilidades motrices se ven mejoradas, puesto que se incita al jugador a utilizar su percepción visual para investigar el entorno. No se puede olvidar el contenido social y motivador de los videojuegos, al ofrecer recompensas e interacción con otros jugadores de manera física o en la red [9].

A la vista de estos beneficios, no es nada descabellado el uso de los videojuegos con fines educativos, ya que se puede aprovechar su componente lúdica para motivar al alumno a adquirir determinados conocimientos. F. I. R. Domínguez [10] destaca la importancia del juego educativo mencionando su componente motivadora, ya que si se crea un software atractivo para el alumno en el que se ensalcen sus logros éste terminará interiorizando las lecciones. La variedad de videojuegos que se pueden crear favorece su uso en muchas asignaturas y determina un intercambio de información directo y ameno entre profesor y alumno.

Pero no hay que olvidar que el videojuego ha de suponer un reto para el alumno, ya que si no hay cierta interacción y dosis de movimiento y acción el efecto puede ser el contrario al que se espera y no se lograrán los objetivos previstos.

En los últimos años se han distribuido videojuegos educativos que pueden ser utilizados por los estudiantes fuera del ámbito escolar como complemento extra. Uno de los más conocidos es *"Brain Training"* [11] comercializado por Nintendo para su consola portátil Nintendo DS. Este videojuego propone al jugador una serie de ejercicios diarios para mantener a pleno rendimiento su cerebro como calculo rápido, memorización de secuencias de números, etc.

Ilustración 8. Brain Training

Otro título interesante es *"Civilization"* [12], juego de estrategia cuyo objetivo es desarrollar una civilización y llevarla a la cumbre tecnológica, económica y de poder a lo largo del tiempo. Este videojuego ayuda al jugador a reflexionar a la hora de tomar decisiones y le ayuda a aprender mejor cómo se desarrolla la historia del mundo y sus pueblos.

Ilustración 9. Civilization IV

Un videojuego educativo más orientado al público infantil es *"Los Lunnis"* de Nintendo DS, basado en el popular programa de televisión para niños que se emitía en TVE, en el que a través de divertidos minijuegos, el niño puede interactuar con sus personajes favoritos del programa y aprender jugando.

Ilustración 10. Los Lunnis, el videojuego

A pesar de todo, los videojuegos educativos no se están implantando como se espera en las aulas. Hay personas que rechazan la inclusión de los videojuegos en la educación de niños y jóvenes. Esto se debe a que en muchos se tratan temas que no son adecuados para ellos, como violencia, sexismo, racismo o erotismo [8, 9]. Por otro lado, se piensa que pueden causar adicción, marginación social o trastornos de personalidad o conducta.

Estos temas generan un sentimiento de rechazo hacia los juegos electrónicos en educadores y padres. No obstante, no hay que censurar la utilización de los videojuegos por su posible contenido, sino controlar el acceso de los niños y jóvenes a determinados juegos. [8, 9, 10]

Otro problema que genera la no utilización de juegos educativos en las aulas es el desconocimiento de las nuevas tecnologías por parte del profesorado. La enseñanza actual continúa enseñando contenidos al alumnado de manera oral y escrita a través de libros y apuntes, pero no recurre normalmente a software que facilite su tarea y la haga más amena de cara a los estudiantes. M. Benito [13] no sólo culpa de este hecho a la baja formación tecnológica de muchos docentes, sino que también atribuye la poca penetración de los videojuegos en las escuelas a las grandes instituciones que dirigen el sistema educativo y que no llevan a cabo medidas para la innovación.

El personal docente ha de estar preparado para asumir todos los adelantos tecnológicos que se vayan produciendo y así enseñar a sus alumnos de manera acorde a las necesidades del momento. Para facilitar un poco la introducción de los videojuegos educativos en sus lecciones, el personal docente puede recurrir a las herramientas de autoría para desarrollar los suyos propios.

Estas aplicaciones se basan en determinados modelos de diseño para confeccionar el que será el videojuego oficial. De entre todos los posibles modelos, se van a tratar dos modelos diferentes:

- **Modelo de juegos adaptativos** [14]. Según este modelo, el juego educativo se compone de actividades que los alumnos han de realizar. Dichas actividades constan de varios objetivos que representan el conocimiento que se ha de alcanzar. Con este modelo se pueden seleccionar las actividades para el alumno en función de sus características, preferencias o edad. De esta manera, se le facilitará el aprendizaje. El creador del juego describirá un modelo del usuario para almacenar sus preferencias y un modelo de juego para detallarlo según dichas preferencias.

- **Modelo conceptual basado en reglas y escenarios** [3]. Se basa en la división en dos submodelos: el modelo asociado a las reglas, que describe cómo el videojuego debería ser jugado, y el modelo asociado al escenario, que define el medio virtual en que el videojuego será jugado, la interfaz provista para interactuar con él y el conjunto de servicios disponibles. El modelo asociado a las reglas se compone de: mecanismos, objetivos, storytelling, socialización, feedback y debriefing. El modelo asociado al escenario consta de caracterización, escenas, contexto, servicios, interfaz e interacción. La combinación de los dos submodelos principales y sus elementos da lugar al videojuego final.

2.3 Herramientas de autoría de diseño de videojuegos

Las herramientas de autoría no son algo nuevo que se haya descubierto hace pocos meses. En este apartado se pretenden mostrar algunos ejemplos de herramientas que son empleadas por docentes para crear sus propios videojuegos:

2.3.1 Erudito

Es una herramienta de creación y monitorización de videojuegos masivos multijugador en línea (MMO) [15]. Los docentes pueden crear el contenido de una determinada lección a través de una metáfora: cada lección viene representada como un mundo en el videojuego; las lecciones se dividen en módulos, (en el juego, el mundo se divide en regiones) que pueden jugarse superando módulos previos o no; los módulos cubren uno o varios conceptos desarrollados por medio de materiales y evaluados a través de preguntas (en el juego, cada concepto es representado con una zona o sector en la que el usuario puede interaccionar con los objetos o resolver acertijos para superar el nivel).

2.3.2 e-Adventure

Es otra herramienta de autoría desarrollada por la Universidad Complutense de Madrid [16, 18]. Consta de un motor propio de ejecución del videojuego y de un editor para poder escribir de manera cómoda el guión.

El motor consta de un sistema de entrada para recopilar datos de los dispositivos de entrada conectados y dirigirlos al controlador adecuado; un HUD, para mostrar los elementos accesibles y las interacciones que se pueden realizar con ellos; el inventario, para mostrar los objetos que tiene el jugador y un menú para cambiar ciertos parámetros del juego. El editor viene equipado con ventanas de previsualización de la escena que el creador del videojuego está desarrollando, controlador de identificadores de objetos, editor gráfico de conversaciones y creación automática de objetos. La inclusión de cámaras es sencilla de llevar a cabo, ya que no es necesario que el usuario introduzca unas determinadas coordenadas para instalarlas.

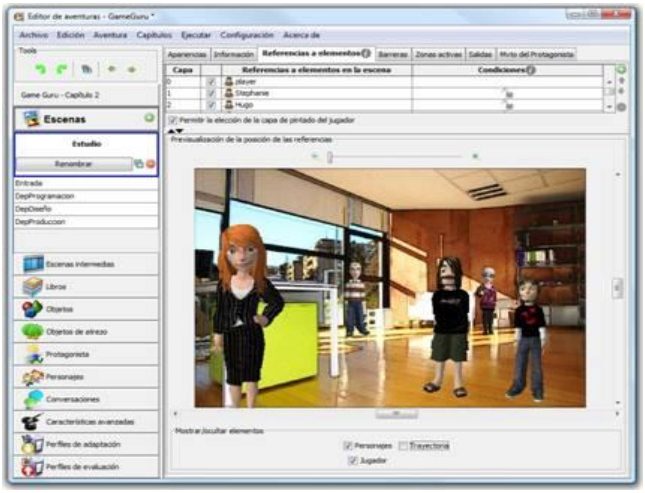


Ilustración 11. Editor de e-Adventure

En un estudio realizado por S. C. Begega y M. Nistal [17], se comparan varias herramientas de autoría (entre ellas e-Adventure) para ver cuál es la más adecuada para crear videojuegos educativos. Tras un exhaustivo examen en el que se crean dos tipos diferentes de videojuegos con e-Adventure y Game Maker, se llega a la conclusión de que el primero es más eficiente puesto que es fácil de utilizar, se necesita un mínimo coste de desarrollo y aporta un gran número de funcionalidades.

2.3.3 Scratch

Es un programa de autoría creado por el Instituto Tecnológico de Massachusetts (MIT) que parte de la idea de encajar ciertos personajes u objetos en un escenario a modo de puzle para que realicen una serie de acciones [19]. Presenta una interfaz intuitiva y amigable, posee un banco de recursos propio y ventanas de previsualización y permite mover los objetos sobre el escenario para posicionarlos en el sitio deseado. Se pueden configurar animaciones, simulaciones o actividades interactivas.

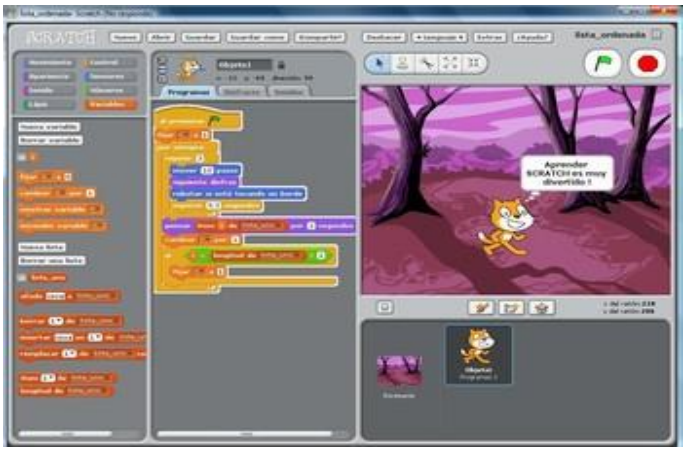


Ilustración 12. Editor de Scratch

2.4 Tecnologías

La herramienta de autoría a desarrollar y eje central de este proyecto tiene como finalidad generar representaciones gráficas de los elementos que se verán incluidos en el videojuego que el docente pretenda crear. La aplicación contará con formularios que el usuario rellenará para configurar las características del videojuego. Una vez establecidos todos los elementos del videojuego, la herramienta generará documentos XML que los representarán. Estos documentos serán interpretados por la plataforma GREP, que a su vez es implementada mediante el motor de videojuegos Unity 3D.

En base a esto, se han de seleccionar las tecnologías más acordes al objetivo de este proyecto para implementarlo. La herramienta será una aplicación web, ya que de este modo resultará más fácil actualizarla en un futuro y mantenerla, y se adaptará a dispositivos portátiles, como las *tablets*.

A continuación, se van a analizar entornos de desarrollo web y otras tecnologías útiles para la construcción de la herramienta de autoría. Después, se compararán las ventajas y desventajas de cada una de las tecnologías y se elegirán las más adecuadas de acuerdo a las necesidades del proyecto.

2.4.1 Análisis de entornos web

Tras un exhaustivo trabajo de recopilación de información, se han escogido seis tipos diferentes de entornos de desarrollo web para ser analizados.

2.4.1.1 jQuery Mobile

jQuery Mobile es un entorno de desarrollo web orientado a la programación de aplicaciones web móviles [20]. Nace a partir de jQuery, biblioteca de JavaScript creada por John Resig, debido a la necesidad de crear aplicaciones diseñadas para dispositivos móviles.

Este *framework* se basa en la filosofía "Escribe menos, consigue más", según la cual se pueden programar aplicaciones web que pueden ejecutarse en cualquier plataforma móvil, ya sea *smartphone*, *tablet*, *laptops* u ordenadores de escritorio.

Ventajas:

- Permite construir aplicaciones compatibles con multitud de plataformas.
- Tiene una curva de aprendizaje baja. Con unos conocimientos básicos de HTML y CSS se pueden construir aplicaciones en poco tiempo.
- Hay a disposición del programador gran cantidad de documentación para su consulta.
- No requiere la utilización de programas especiales.
- El *framework* es simple y tiene un tamaño reducido.
- Es compatible con HTML5.
- Posibilita la creación de plantillas personalizables.
- Cuenta con editor visual.
- Es *open source*.
- Facilidad de instalación. Se descarga el código fuente, se almacena dentro del proyecto y se incluye una referencia en las páginas HTML de la aplicación.

Desventajas:

- Ofrece muchas funciones, pero no son fáciles de personalizar.
- Presenta un aspecto visual estandarizado.
- La carga de la página se ve ralentizada por la invocación del archivo jQuery.
- El manejo de CSS resulta complejo por la variedad de clases disponibles.
- Tiene pocas plantillas prediseñadas.

2.4.1.2 Bootstrap

Bootstrap es un *framework* desarrollado por Twitter [21]. Combina CSS y JavaScript para facilitar la labor de programación de una aplicación web, sobre todo en lo que se refiere al *front-end*.

Ventajas:

- Permite la creación de aplicaciones que se adapten a los diferentes navegadores, tanto las de escritorio como las que se ejecutan en *tablets* u otros dispositivos móviles.
- Utiliza LESS, un pre-compilador de CSS, que facilita las labores de diseño, y estándares CSS3 y HTML5.
- Se integra de manera sencilla en el proyecto de la aplicación.
- Dispone de *layouts* predefinidos con distinto número de columna y diseño.
- Tiene una enorme comunidad de desarrolladores en la que compartir dudas.
- Es *open source*.
- Curva de aprendizaje baja.

Desventajas:

- Es necesario familiarizarse a la estructura y nomenclatura del framework.
- El diseño gráfico se debe adaptar a las 12 columnas que vienen ya determinadas.
- Los componentes añadidos externos a Bootstrap deben ser personalizados con CSS previamente.

- Tamaño de *framework* elevado (más de 100 KB).
- Problemas de rendimiento al cargar la página cuando se invocan los ficheros del *framework*.

2.4.1.3 Foundation

Foundation es un entorno de desarrollo web creado por Zurb, equipo de diseñadores estadounidenses que presta servicios tecnológicos a otras empresas [22]. Actualmente se encuentra en su versión 4.0, lanzada el mes de marzo de 2013. Al igual que Bootstrap, está orientado al diseño del *front-end* de las aplicaciones web.

Ventajas:

- Es compatible con cualquier dispositivo y cuenta con un soporte de presentación para facilitar la creación de diseños.
- Permite incluir elementos útiles que permitan, por ejemplo, ocultar o mostrar cosas de acuerdo a la orientación de la pantalla o el tamaño.
- Nomenclatura sencilla.
- Incluye modelos de formularios, botones, elementos de interfaz, etc.
- Es *open source*.
- Altamente documentado.
- Utiliza Sass, un pre-compilador de CSS, para facilitar las labores de diseño.

Desventajas:

- Tamaño de *framework* elevado (más de 100 KB).
- Pequeños problemas de rendimiento al cargar la página cuando se invocan los ficheros del *framework*.

2.4.1.4 Zend

Zend es un *framework* de desarrollo web para PHP creado por Zend Technologies, empresa que provee herramientas a diseñadores web para ayudarles en su trabajo [23].

Ventajas:

- Permite desarrollar aplicaciones web robustas.
- Facilita el mantenimiento de las aplicaciones.
- Ofrece seguridad frente a diversos ataques.
- Es *open source*.
- Cuenta con una buena comunidad de desarrolladores.

Desventajas:

- Curva de aprendizaje alta.
- Difícil instalación, pues el entorno necesita ser configurado para poder funcionar correctamente.

- Necesita un servidor dedicado para aprovechar al máximo las funciones con las que cuenta.
- Cualquier cambio ha de ser gestionado a través de la línea de comandos del servidor dedicado.

2.4.1.5 CodeIgniter

CodeIgniter es un *framework* de desarrollo web ideado por EllisLab, Inc. ideado para soportar la creación de cualquier aplicación web bajo PHP [24].

Ventajas:

- Ofrece un gran rendimiento.
- Es versátil, ya que es capaz de trabajar con la mayoría de entornos o servidores.
- Es compatible con varias versiones de PHP (desde la 4.3.2 a la 5.3.0).
- No requiere tanta configuración como necesita Zend.
- No necesita la línea de comandos para generar las aplicaciones.
- Tiene buena documentación.
- Es *open source*.
- El tamaño del *framework* no es muy elevado (1 MB), teniendo en cuenta la alta cantidad de librerías que posee.

Desventajas:

- Curva de aprendizaje inferior a Zend, pero implica familiarizarse con el lenguaje.
- No tiene sistema de *layouts* o plantillas.

2.4.1.6 HTML Kickstart

HTML Kickstart es un entorno de desarrollo web creado por 99Lime útil para diseñar el front-end de las aplicaciones web [25]. Como tal, se basa en HTML, CSS y JavaScript como lenguajes de marcado, diseño e interacción con las páginas de la aplicación.

Ventajas:

- Es compatible con bastantes navegadores, incluidos los más usados (Google Chrome, Firefox, Internet Explorer 8, etc.).
- Es *open source*.
- Compatible con HTML5.
- Ofrece multitud de elementos web predefinidos, como botones, sliders, iconos, menús, formularios, etc.
- Curva de aprendizaje baja.
- Fácil instalación. Se descarga el código fuente, se almacena dentro del proyecto y se incluye una referencia en las páginas HTML de la aplicación.

Desventajas:

- El diseño gráfico se debe adaptar a las 12 columnas que vienen ya determinadas. Pequeños problemas de rendimiento al cargar la página cuando se invocan los ficheros del *framework*.

2.4.2 Comparación de alternativas y elección del entorno de desarrollo

Tras analizar las características de los posibles entornos de desarrollo web a utilizar, se pretende a continuación compararlos con vistas a elegir el más apropiado para implementar la herramienta de autoría.

En la siguiente tabla se muestran las tecnologías candidatas evaluadas en base a unos parámetros escogidos:

	JQuery Mobile	Bootstrap	Foundation	Zend	CodeIgniter	HTML Kickstart
Aspecto adaptado para tablets	**Sí**	**Sí**	**Sí**	**No**	**No**	**Sí**
Curva de aprendizaje	**Baja**	**Baja**	**Baja**	**Alta**	**Alta**	**Baja**
Tratamiento de formularios	**Sí**	**Sí**	**Sí**	**Sí**	**Sí**	**Sí**
Open Source	**Sí**	**Sí**	**Sí**	**Sí**	**Sí**	**Sí**
Volumen de documentación	**Alto**	**Alto**	**Alto**	**Alto**	**Alto**	**Medio**
Lenguajes aceptados	**HTML, CSS, JavaScript**	**HTML, CSS, JavaScript**	**HTML, CSS, JavaScript**	**PHP**	**PHP**	**HTML, CSS, JavaScript**
Rendimiento	**Regular (pequeños problemas de carga)**	**Regular (pequeños problemas de carga)**	**Regular (pequeños problemas de carga)**	**Bueno**	**Bueno**	**Bueno**
Tamaño	**~32KB**	**~130KB**	**~180KB**	**~3MB**	**~1MB**	**~64KB**
Facilidad de instalación	**Sí**	**Sí**	**Sí**	**No**	**No**	**Sí**

Tabla 1. Comparativa de tecnologías de desarrollo web

Uno de los requisitos imprescindibles de la herramienta es que esté adaptada a tablets y su resolución. Observando la tabla comparativa se aprecia que ni Zend ni CodeIgniter posibilitan este requisito, por lo que quedan inmediatamente descartados como candidatos. Aunque no hubiera sido así, no habrían sido escogidos puesto que tienen una curva de aprendizaje y un tamaño elevados.

HTML Kickstart presenta una curva de aprendizaje baja pero al ser una tecnología de aparición reciente no hay mucha documentación disponible, por lo que si surge algún

problema durante la implementación sería complicado encontrar una solución. Por esta razón, esta tecnología ha sido desechada.

jQuery Mobile es la tecnología que ocupa un menor tamaño, pero el manejo de CSS no es sencillo debido a su variedad de clases disponibles. Por otro lado, presenta un aspecto visual demasiado estándar.

Foundation y Bootstrap son bastante similares, pero se ha decidido optar por Bootstrap porque su tamaño es inferior y sus componentes y clases CSS se adaptan mejor a la herramienta que se pretende conseguir.

2.4.3 eXtensible Markup Language (XML)

eXtensive Markup Language, más conocido como XML, es un lenguaje de marcado, desarrollado por el World Wide Web Consortium (W3C) [26], que juega un papel fundamental en el intercambio de una gran variedad de datos. Es muy parecido al lenguaje HTML, pero se diferencian en sus funciones principales. Así como HTML muestra datos, XML los describe para que puedan ser intercambiados y leídos por diferentes aplicaciones. Un documento XML se compone de elementos definidos entre etiquetas que a su vez pueden contener otros elementos.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <RootEmpleados>
  - <Empleados>
      <IdEmpleado>1</IdEmpleado>
      <Name>Arbis Percy Reyes Paredes</Name>
      <Direccion>Urb. Luis albretch Mz. M #103</Direccion>
      <Ciudad>Trujillo</Ciudad>
      <Pais>Peru</Pais>
      <ocupacion>Ing. de Software</ocupacion>
    </Empleados>
  - <Empleados>
      <IdEmpleado>2</IdEmpleado>
      <Name>Milagros Quiroz Torrejon</Name>
      <Direccion>Urb. Los Pinos Mz. G #311</Direccion>
      <Ciudad>Malaga</Ciudad>
      <Pais>Espana</Pais>
      <ocupacion>Ing. de Sistemas Informaticos</ocupacion>
    </Empleados>
```

Ilustración 13. Ejemplo de documento XML

3. Gestión del proyecto

En esta sección se expone toda la información relativa a la gestión del proyecto. En primer lugar, se presentará el modelo de ciclo de vida que se ha utilizado y las fases del desarrollo. Posteriormente, se mostrará la planificación inicial del proyecto y se adjuntarán los costes derivados de la consecución del proyecto.

3.1 Modelo de ciclo de vida

Tras estudiar los modelos de ciclo de vida disponibles y de acuerdo a las necesidades que el presente proyecto exigía, se ha decidido que el modelo elegido sea el de ciclo de vida en cascada con retroalimentación.

Este modelo se compone de una serie de etapas bien diferenciadas que se realizan secuencialmente, no pudiéndose comenzar una nueva etapa hasta que finaliza la anterior. Esto quiere decir que la salida de una etapa corresponde con la entrada de la siguiente. No obstante, goza de una peculiaridad que le confiere cierto atractivo: se trata de un modelo retroalimentado, lo que quiere decir que permite efectuar cambios en cualquier etapa del desarrollo y añadirlos en fases posteriores sin enturbiar el correcto avance de un proyecto.

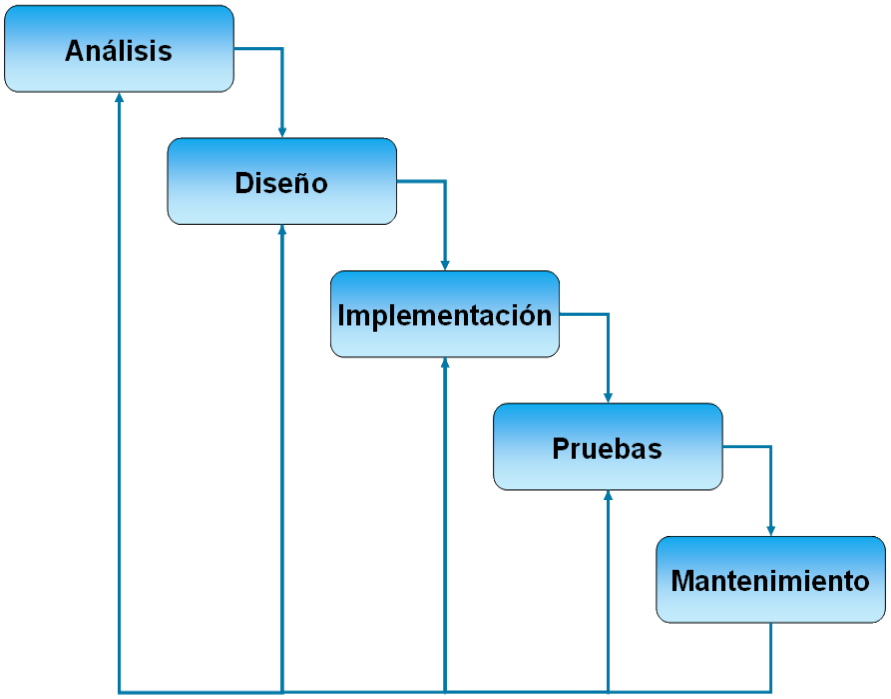


Ilustración 14. Ciclo de vida en cascada con retroalimentación

Este modelo es sencillo y facilita la gestión del proyecto en cada una de las fases. No obstante, precisa que los requisitos estén bien delimitados desde el momento en que se efectúa la fase de análisis.

3.2 Fases del desarrollo

En esta subsección se van a detallar las fases que se han aplicado en el proceso de desarrollo del proyecto:

- **Fase de investigación**. Comprende todo el proceso de búsqueda de información relacionada con el proyecto, desde los diferentes estudios existentes del problema a remediar, software creado, etc. hasta las tecnologías que ayudasen a hacerlo realidad.

- **Fase de análisis**. En ella se identificaron los principales requisitos que la herramienta ha de cumplir para su correcto funcionamiento.

- **Fase de diseño**. En esta fase se estableció una solución apropiada para implementar todos los requisitos determinados en la fase de análisis.

- **Fase de implementación**. Tras la fase de diseño tuvo lugar la etapa de programación de la herramienta, en la que se siguieron las premisas especificadas para obtener el producto final usable.

- **Fase de pruebas**. Una vez implementada la herramienta, se dedicó un tiempo a crear baterías de pruebas que cubrieran los aspectos más destacables de su funcionalidad.

- **Fase de documentación**. En esta fase se preparó el presente documento con vistas a obtener una perfecta comprensión acerca del proyecto y de todos los puntos relevantes relacionados con él.

3.3 Planificación

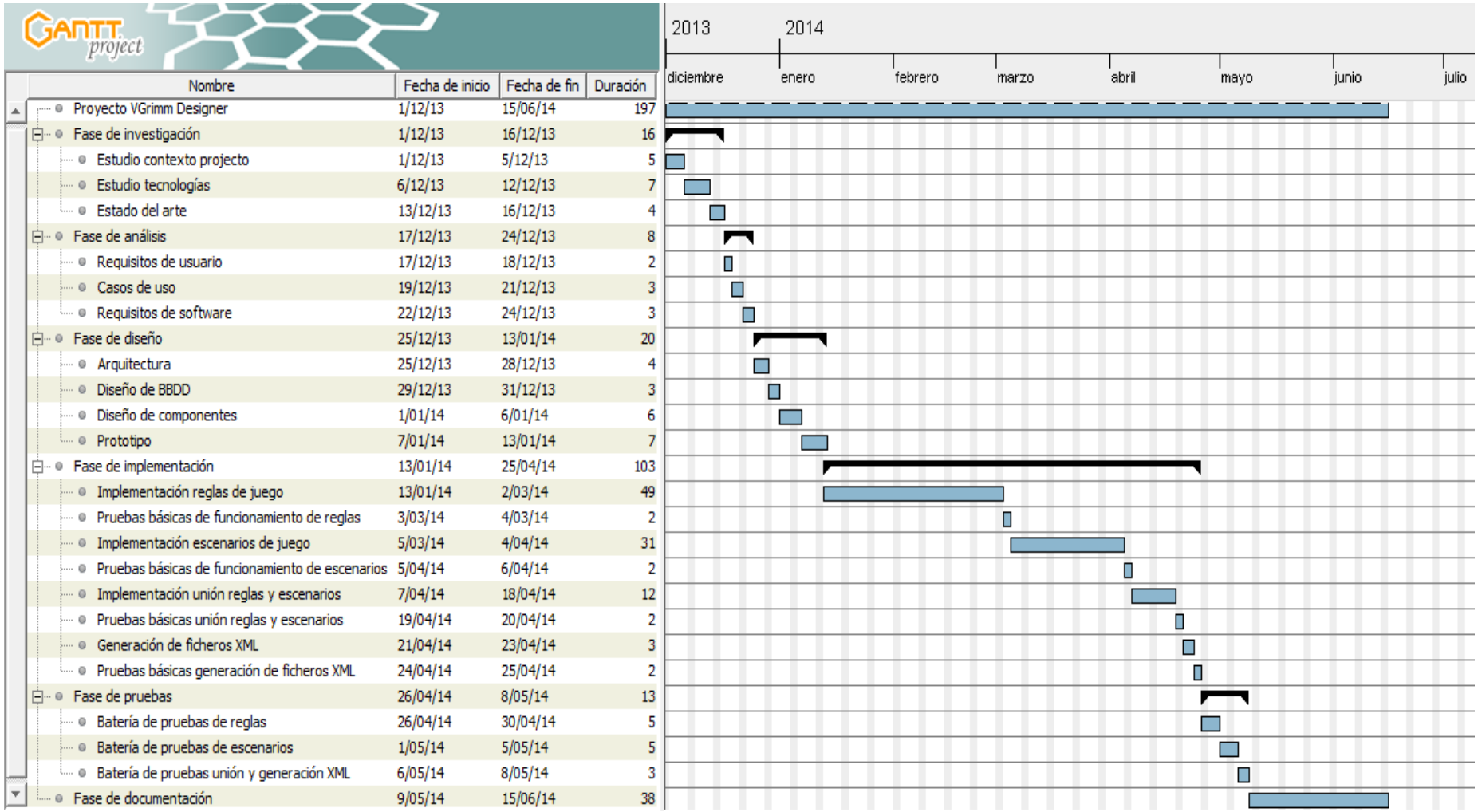
En un proyecto es importante estimar la cantidad de tiempo que va a conllevar su desarrollo. Para ello, es preciso identificar cada una de las tareas que han de realizarse y establecer unos periodos de inicio y finalización de todas ellas. En esta subsección se incluye la planificación inicial del proyecto, su duración y las fases que lo componen.

3.3.1 Planificación inicial

El proyecto tiene una duración aproximada de 7 meses (197 días). Su fecha de inicio se corresponde con el 1 de diciembre, mientras que la fecha de fin se sitúa en torno al 15 de junio. Si suponemos una media de trabajo de 4 horas al día incluyendo fines de semana y festivos, la suma de horas semanales es equivalente a 28 horas, lo que daría lugar a un total estimado de 788 horas dedicadas al proyecto desde su inicio hasta su fin.

La jornada laboral es reducida puesto que el desarrollador principal del proyecto es el responsable del presente documento y no puede dedicar más tiempo al compaginar el proyecto con un trabajo a media jornada. Se han decidido incluir fines de semana y días festivos porque en esa franja horaria el desarrollador dispone de una mayor cantidad de tiempo para trabajar en el proyecto en caso de producirse algún tipo de retraso respecto a la planificación.

A continuación, se muestra el desglose de tareas a cumplir en el proyecto junto con su duración aproximada en forma de diagrama de Gantt:



Nombre	Fecha de inicio	Fecha de fin	Duración
Proyecto VGrimm Designer	1/12/13	15/06/14	197
Fase de investigación	1/12/13	16/12/13	16
Estudio contexto projecto	1/12/13	5/12/13	5
Estudio tecnologías	6/12/13	12/12/13	7
Estado del arte	13/12/13	16/12/13	4
Fase de análisis	17/12/13	24/12/13	8
Requisitos de usuario	17/12/13	18/12/13	2
Casos de uso	19/12/13	21/12/13	3
Requisitos de software	22/12/13	24/12/13	3
Fase de diseño	25/12/13	13/01/14	20
Arquitectura	25/12/13	28/12/13	4
Diseño de BBDD	29/12/13	31/12/13	3
Diseño de componentes	1/01/14	6/01/14	6
Prototipo	7/01/14	13/01/14	7
Fase de implementación	13/01/14	25/04/14	103
Implementación reglas de juego	13/01/14	2/03/14	49
Pruebas básicas de funcionamiento de reglas	3/03/14	4/03/14	2
Implementación escenarios de juego	5/03/14	4/04/14	31
Pruebas básicas de funcionamiento de escenarios	5/04/14	6/04/14	2
Implementación unión reglas y escenarios	7/04/14	18/04/14	12
Pruebas básicas unión reglas y escenarios	19/04/14	20/04/14	2
Generación de ficheros XML	21/04/14	23/04/14	3
Pruebas básicas generación de ficheros XML	24/04/14	25/04/14	2
Fase de pruebas	26/04/14	8/05/14	13
Batería de pruebas de reglas	26/04/14	30/04/14	5
Batería de pruebas de escenarios	1/05/14	5/05/14	5
Batería de pruebas unión y generación XML	6/05/14	8/05/14	3
Fase de documentación	9/05/14	15/06/14	38

Ilustración 15. Diagrama de Gantt de la planificación inicial

Como se puede ver en el diagrama, cada fase del desarrollo está dividida en diferentes tareas con una duración asociada. La fase de implementación ocupa más del 50% de la vida del proyecto porque en ella es concebida la herramienta de autoría como tal. En esta fase se identifican cuatro tareas principales, correspondientes a la programación de la funcionalidad de la herramienta: implementación de reglas de juego, de escenarios, unión de reglas y escenarios y generación de ficheros XML. Al finalizar cada tarea, se realizan unas pequeñas pruebas para comprobar de una manera muy básica que todo está en orden. Las pruebas avanzadas se llevan a cabo en la fase de pruebas.

3.4 Presupuesto

En esta subsección se van a detallar los costes derivados del desarrollo del proyecto, ya que se necesitan tanto recursos humanos como recursos físicos para su consecución.

3.4.1 Recursos humanos

Tal y como ha quedado reflejado en la subsección de Planificación, la persona sobre la que recae el mayor peso del proyecto es el alumno, mientras que las labores de apoyo son responsabilidad de su tutor. No obstante, este último interpreta en realidad el rol de cliente, por lo que no forma parte del personal de recursos humanos y no representa ningún coste dentro del proyecto.

A falta de un mayor número de personal, el alumno ejecuta todos los roles implicados en el desarrollo del proyecto, por lo que se van a calcular los costes originarios de cada uno de ellos.

Rol	Horas dedicadas	Salario/hora (€/hora)	Total (€)	Coste Seguridad Social	Total + Coste Seguridad Social
Jefe de proyecto	**60**	**38 €**	**2280 €**	**538,08 €**	**2818,08 €**
Analista	**32**	**29 €**	**928 €**	**219,008 €**	**1147,008 €**
Diseñador	**80**	**27 €**	**2160 €**	**509,76 €**	**2669,76 €**
Programador	**380**	**22 €**	**8360 €**	**1972,96 €**	**10332,96 €**
Responsable de pruebas	**84**	**27 €**	**2268 €**	**535,248 €**	**2803,248**
Documentador	**152**	**19 €**	**2888 €**	**681,568 €**	**3569,568 €**
TOTAL	**-**	**-**	**-**	**-**	**23340,624 €**

Tabla 2. Costes de recursos humanos

Las horas dedicadas se han obtenido a partir de la planificación inicial teniendo en cuenta que cada día el alumno dedica 4 horas al proyecto independientemente del rol que ejecute en cada momento. Sobre el coste total de cada rol se ha aplicado la cuota empresarial a la Seguridad Social del 23,6 % correspondiente a contingencias comunes.

Tras sumar todos los costes de recursos humanos, el total asciende a VEINTITRÉS MIL TRESCIENTOS CUARENTA EUROS CON SESENTA Y DOS CÉNTIMOS (23340,62 €).

3.4.2 Recursos físicos

Además de personal cualificado, el proyecto va a necesitar una serie de recursos físicos, los cuales se dividen en hardware, software, material de oficina y costes fijos.

3.4.2.1 Hardware

El hardware incluye el equipo o equipos que se van a utilizar en el proyecto y otro material como impresoras.

Hardware	Precio producto (€)	Duración del proyecto (meses)	Periodo de amortización (años)	Precio/unid. (€)	Cantidad	Coste total
Portátil HP Pavilion DV6-1430SS Intel Core Duo	**699 €**	**7 meses**	**4 años**	**101,94 €**	**1**	**101,94 €**
Impresora HP Deskjet F4580	**75 €**	**7 meses**	**4 años**	**10,94 €**	**1**	**10,94 €**
Ratón óptico	**10 €**	**7 meses**	**4 años**	**1,46 €**	**1**	**1,46 €**
TOTAL	**-**	**-**	**-**	**-**	**-**	**114,34 €**

Tabla 3. Costes de hardware

Al precio de cada producto se le ha aplicado una amortización de 4 años, lo que ha derivado en un precio unitario obtenido a partir de la siguiente expresión:
Precio/unid. = (Precio producto * Duración del proyecto) / Periodo de amortización.

3.4.2.2 Software

El software incluye todos aquellos programas o entornos necesarios para llevar a cabo el proyecto.

Software	Precio producto (€)	Duración del proyecto (meses)	Periodo de amortización (años)	Precio/unid. (€)	Cantidad	Coste total
Windows 7 Home Premium	**0 €**	**7 meses**	**4 años**	**0 €**	**1**	**0 €**
Microsoft Office 2010	**150 €**	**7 meses**	**4 años**	**21,875 €**	**1**	**21,875 €**
IBM Rational Software Architect	**10954,13 €**	**7 meses**	**1 año**	**6389,91 €**	**1**	**6389,91 €**
Axure RP Pro 7	**215 €**	**7 meses**	**4 años**	**31,35 €**	**1**	**31,35 €**
TOTAL	**-**	**-**	**-**	**-**	**-**	**6443,13 €**

Tabla 4. Costes de software

Al precio de cada producto se le ha aplicado una amortización de 4 años, lo que ha derivado en un precio unitario obtenido a partir de la siguiente expresión:
Precio/unid. = (Precio producto * Duración del proyecto) / Periodo de amortización.
La licencia de Windows 7 Home Premium viene incluida con la compra del equipo portátil.

3.4.2.3 Material de oficina

Material	Precio producto (€)	Cantidad	Coste total
Pack cartucho tinta negra + color	**34,99 €**	**4**	**139,96 €**
Pack 500 folios DIN-A4	**2,35 €**	**3**	**7,05 €**
Pack de material de oficina	**30 €**	**1**	**30 €**
Bote de imprevistos	**200 €**	**1**	**200 €**
TOTAL	**-**	**-**	**377,01 €**

Tabla 5. Costes de material de oficina

3.4.2.4 Costes fijos

Producto	Precio mensual (€)	Duración del proyecto (meses)	Coste total
Internet	**32 €**	**7**	**224 €**
Luz	**30 €**	**7**	**210 €**
TOTAL	**-**	**-**	**434 €**

Tabla 6. Costes fijos

3.4.2.5 Coste total

Concepto	Coste
Hardware	**114,34 €**
Software	**6443,13 €**
Material de oficina	**377,01 €**
Costes fijos	**434 €**
TOTAL	**7368,48 €**

Tabla 7. Coste recursos físicos

Tras sumar todos los costes de recursos físicos, el total asciende a SIETE MIL TRESCIENTOS SESENTA Y OCHO EUROS CON CUARENTA Y OCHO CÉNTIMOS (7368,48 €).

3.4.3 Presupuesto final

Concepto	Coste
Recursos humanos	**23340,62 €**
Recursos físicos	**7368,48 €**
TOTAL	**30709,10 €**

Tabla 8. Presupuesto final

Después de sumar todos los costes de recursos humanos y recursos físicos, el presupuesto final asciende a TREINTA MIL SETECIENTOS NUEVE EUROS CON DIEZ CÉNTIMOS (30709,10 €) IVA incluido.

3.5 Entorno socio-económico

La industria del videojuego ha evolucionado de una manera extraordinaria desde que se convirtió en uno de los pilares del ocio de las personas. Desde el nacimiento del primer videojuego en 1952 hasta el día de hoy los avances tecnológicos han permitido que esta industria crezca hasta convertirse en un negocio bastante lucrativo.

En sus comienzos, los videojuegos eran concebidos por una única persona o por un pequeño grupo de desarrolladores que plasmaban sus ideas frente al ordenador. En la actualidad, son creados por un equipo multidisciplinar de ingenieros que utilizan la tecnología más puntera para dar lugar a productos con una gran calidad gráfica y variedad de posibilidades de juego. Este importante cambio se debe no sólo a la evolución de la tecnología sino también al interés cada vez mayor en la industria del videojuego por parte de desarrolladores y usuarios.

En un momento en que la recesión económica y su evolución acaparan el interés mediático, muchas personas optan por evadirse de la realidad y el ocio electrónico es un potencial factor de distracción. Las principales compañías de la industria del videojuego son conscientes de esto y dedican bastante tiempo a la creación de campañas de marketing, conferencias y eventos que acerquen sus productos disponibles a los consumidores a la vez que desarrollan otros nuevos.

No obstante, la aparición del mercado de los videojuegos orientados a dispositivos portátiles ha provocado que el dominio de la industria no recaiga solamente en las grandes compañías. En tiempos de crisis, la posibilidad de probar videojuegos de manera gratuita, rápida y cómoda en dispositivos móviles como *smartphones* o *tablets* supone un importante foco de atracción de numerosos consumidores que buscan una alternativa de ocio fácil. Por otro lado, los usuarios habituales de videojuegos demandan productos de mayor calidad y realismo.

Las demandas por parte de los usuarios y la creciente competencia han favorecido las investigaciones dentro de las empresas del sector para mejorar la calidad y las

prestaciones de sus productos. La industria de los videojuegos proporciona grandes beneficios pero es preciso mantenerse a la vanguardia de las tecnologías y satisfacer las necesidades de los usuarios, lo que genera un alto coste en investigaciones.

A día de hoy es posible concebir una unión entre videojuegos y educación, ya que los niños aprenden rápido a usar los dispositivos electrónicos a su alcance, está demostrado que los videojuegos aportan beneficios educativos y existe una importante cantidad de investigadores y desarrolladores dispuestos a identificar y proporcionar las pautas y los videojuegos necesarios.

3.6 Marco regulador

Según la *Ley Orgánica 15/1999 de 13 de diciembre de Protección de Datos de Carácter Personal* (LOPD) se entiende por datos de carácter personal "cualquier información concerniente a personas físicas identificadas o identificables". La herramienta de autoría desarrollada en este proyecto no utiliza información personal de sus usuarios, sino datos que ellos mismos proporcionan para configurar sus propios videojuegos. Por lo tanto, la herramienta queda fuera del ámbito de aplicación de dicha ley.

Para la implementación de la herramienta se va a utilizar el entorno de programación Rational Software Architect de IBM que requiere una licencia para su uso. En el presupuesto se ha indicado el precio de una licencia estándar de este software. No obstante, la universidad ha proporcionado al alumno una copia del mencionado software con licencia incluida, por lo que las restricciones legales derivadas de este acuerdo con la empresa creadora del software no afectan directamente al creador de la herramienta de autoría.

Del mismo modo, se ha incluido en el presupuesto el precio de una licencia estándar del software de diseño Axure RP, pero en realidad se ha utilizado una versión gratuita con menos funcionalidades, lo que evita las responsabilidades legales pertinentes con la empresa creadora del producto. El resto de tecnologías utilizadas como Bootstrap o Java no necesitan de ninguna clase de licencia para su uso.

Asimismo, el presente documento y la herramienta de autoría se encuentran amparados por las leyes de propiedad intelectual españolas: *Ley 23/2006* y *Real Decreto Legislativo 1/1996, de 12 de abril*.

4. Análisis

En esta sección se realiza un análisis exhaustivo del sistema que se va a implementar. En primer lugar, se dará una descripción general del sistema en la que se tratará su funcionalidad. Es importante dejar claro el alcance de la herramienta desde la etapa de análisis para poder definir unos requisitos adecuados.

Tras ello, se incluirá el catálogo de requisitos de usuario obtenido a partir de las necesidades del cliente, Telmo Zarraonandia, que se reunió con el alumno para establecer con él las diferentes funcionalidades de la herramienta. Después de los requisitos de usuario, se adjuntan los diferentes casos de uso con los que se llevará a cabo el análisis propiamente dicho de la funcionalidad de la herramienta teniendo en cuenta los mencionados requisitos de usuario.

A continuación, se presentará el catálogo de requisitos de software, que tratan a un nivel más bajo las capacidades o restricciones detectadas con los requisitos de usuario. Finalmente, se añadirán las correspondientes matrices de trazabilidad. Dichas matrices aseguran la consistencia de los requisitos y permiten una monitorización rápida de las relaciones entre requisitos de software, de usuario y casos de uso.

4.1 Descripción general del sistema

El cliente necesita una herramienta de autoría que permita al personal docente el diseño conceptual de sus propios videojuegos de una manera intuitiva y sencilla. Esta herramienta va a ser accesible tanto desde un equipo de escritorio como desde un dispositivo portátil (*tablets*) a través de la Web.

Como se ha especificado en la subsección 1.2 (Objetivo), las directrices que el usuario va a seguir para diseñar su videojuego se basan en el modelo GREM, según el cual los componentes de un videojuego se engloban en dos submodelos distintos: el modelo de reglas de juego y el modelo de escenarios. En base a ello, la herramienta permitirá la creación de elementos de reglas de juego y de escenarios a través de una serie de formularios que el usuario tendrá que rellenar. Estas dos funcionalidades se detallan en las siguientes líneas:

- **Creación de elementos de reglas de juego**. El usuario navegará a través de los diferentes niveles que proporciona el modelo de reglas de juego para completar la información necesaria para obtener un conjunto de reglas. Por cada elemento de cada nivel se presentará un formulario que el usuario cumplimentará para crearlo y darle persistencia. Una vez concebido cualquier elemento, la herramienta posibilitará la edición del mismo para modificar los datos que se deseen. Adicionalmente, se aportarán formularios de consulta para localizar elementos ya creados.

  Después de que todos los elementos estén preparados, se ofrecerá un formulario adicional en el que seleccionar aquellos que se deseen incluir para generar el conjunto de reglas final.

Tras evaluar el grueso del proyecto y la duración estimada del mismo, se ha determinado suspender la implementación de los siguientes elementos de reglas de juego: Debriefing, Socialización, Recompensa y Persistencia. El cliente ha aprobado la decisión puesto que su inclusión en el proyecto conllevaría un mayor coste en tiempo y esfuerzo y su exclusión del mismo no entorpecería el objetivo principal de la herramienta. Por otro lado, la plataforma GREP aún no incluye los módulos para interpretar estos componentes.

- **Creación de elementos de escenario**. Presentará un funcionamiento prácticamente similar al de reglas de juego. El usuario tendrá que rellenar un formulario por cada elemento de cada nivel y guardarlo para utilizarlo después.

  Del mismo modo que con las reglas, la herramienta aportará un formulario extra en el que se podrán agregar los diferentes elementos creados y así obtener un escenario completo.

La idea principal del modelo GREM radica en la unión de reglas y escenarios para dar vida a un videojuego. A partir de esta idea deriva el objetivo de la siguiente funcionalidad del sistema, que consiste en la "mezcla" de un conjunto de reglas y un escenario creados por el usuario. Para ello, el sistema aportará una sección en la que visualizar los elementos de cada uno de ellos y unirlos para generar el diseño final del videojuego.

Como se ha indicado en la subsección 1.2, el diseño final viene representado por una serie de documentos XML. Estos documentos contendrán la unión resultante del conjunto de reglas y el escenario escogidos por el usuario y cuya generación se llevará a cabo automáticamente por parte de la herramienta en el momento en que dicho usuario lo autorice y sin que él tenga que participar en el proceso.

En el diseño se definen los elementos que aparecerán en el videojuego educativo y sus características, pero no por ejemplo qué posición ocuparán dentro del escenario virtual 3D. Estos detalles se indican en el editor que proporciona la plataforma GREP una vez interpretado el diseño. Por lo tanto, el proceso de creación del videojuego no termina en el momento en que los ficheros XML pasan a la plataforma GREP, sino que es preciso que el usuario interactúe con ella para poner a punto su videojuego educativo.

El aspecto del editor GREP es el siguiente:

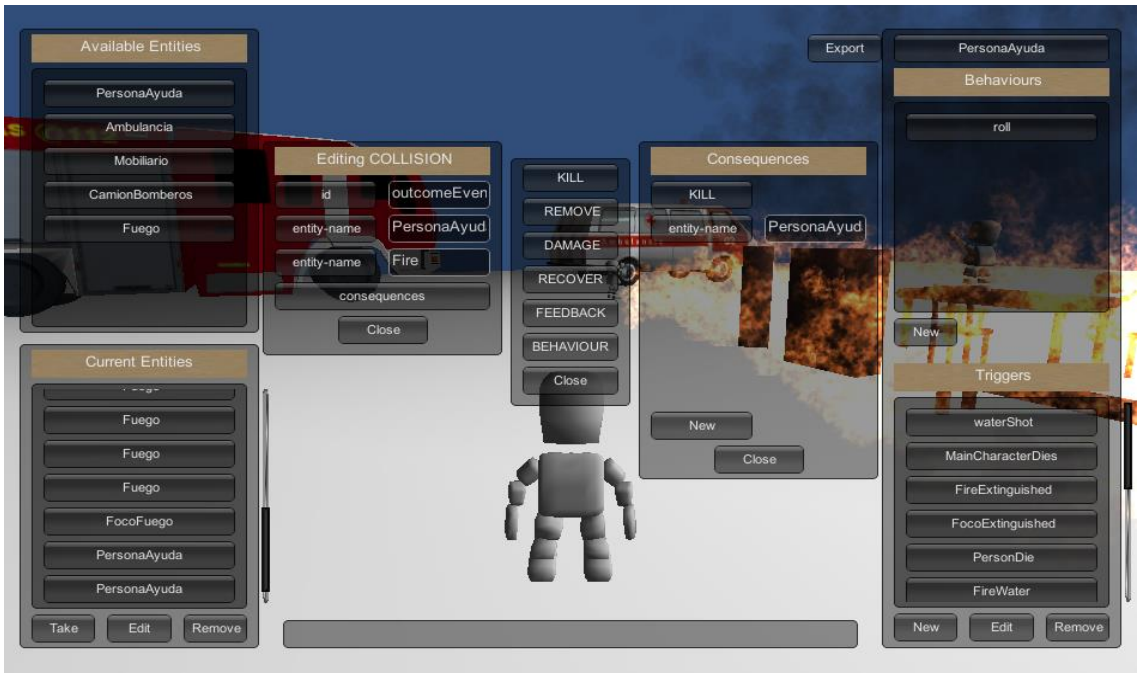


Ilustración 16. Editor GREP

4.2 Catálogo de requisitos de usuario

Los requisitos de usuario expresan deseos o necesidades del cliente y deben ser escritos en un lenguaje comprensible para él. Con ellos el cliente deja clara la funcionalidad que quiere que presente el sistema solicitado. Estos requisitos pueden ser de los siguientes tipos:

- **Requisitos de capacidad**. Especifican las capacidades que debe poseer el sistema para satisfacer las necesidades del cliente.

- **Requisitos de restricción**. Definen restricciones acerca de cómo debe ser construido y operado el sistema.

Para una correcta definición y comprensión de los requisitos, es necesario que estos cumplan una serie de propiedades, que son:

- **Completitud**. La integridad de los requisitos no debe ser comprometida, es decir, no ha de faltar ninguno de los requisitos ni tampoco detalles en la especificación de cada uno.

- **Consistencia**. Los requisitos no deben ser contradictorios. Una buena organización evita la aparición de contradicciones.

- **Claridad**. Ningún requisito puede ser especificado de manera ambigua.

- **Verificabilidad**. Cualquier requisito ha de ofrecer la posibilidad de ser comprobado de una manera sencilla y posible.

- **Variabilidad**. Los requisitos deben propiciar su modificación en caso de que sea necesario de una manera consistente y fácil.

Con vistas a facilitar la lectura y organización de los requisitos, cada uno de ellos va a ser incluido en una tabla como la siguiente:

Identificador			
Nombre			
Descripción			
Fuente		**Prioridad**	
Necesidad		**Estabilidad**	

Tabla 9. Tabla de requisitos modelo

Los atributos de la tabla son explicados a continuación:

- **Identificador**. Permite registrar de manera única un requisito de acuerdo con una determinada nomenclatura: **RUX-YY**. **RU** hace referencia a un requisito de usuario. **X** puede tomar dos valores distintos: 'C' si se trata de un requisito de capacidad y 'R' a uno de restricción. **YY** es un valor numérico que oscila entre 01 y 99.

- **Nombre**. Descripción breve del requisito.

- **Descripción**. Explicación detallada y clara del objetivo del requisito

- **Fuente**. Indica el origen del requisito (cliente, analista, etc.).

- **Prioridad**. Especifica el grado de importancia de cada requisito. Toma tres posibles valores: *Alta*, *Media* y *Baja*. Un requisito con una prioridad alta es vital en el proyecto, por lo que su inclusión es obligatoria. Otro con una prioridad media es menos relevante que el anterior, pero su implementación en el proyecto es esencial para mantener un grado de calidad en el mismo. Finalmente, un requisito de prioridad baja no afecta a la calidad del proyecto, por lo que puede agregarse una vez añadidos todos aquellos de prioridad superior.

- **Necesidad**. Muestra el grado de obligatoriedad de cada requisito. Toma tres valores diferentes: *Esencial*, *Deseable* y *Opcional*. Un requisito esencial es aquel que ha de cumplirse sí o sí en el proyecto. Un requisito deseable no es obligatorio, pero su presencia es aconsejable. Por último, un requisito opcional es aquel que si no se cumple no afecta al rendimiento normal del sistema.

- **Estabilidad**. Define el grado de modificabilidad de cada requisito. Puede adquirir los siguientes valores: *Alta*, *Media* y *Baja*. Un requisito altamente estable no va a ser modificado a lo largo del ciclo de vida del proyecto. Un

requisito con un índice de estabilidad medio presenta una probabilidad media de ser modificado de acuerdo a los cambios que se presenten. Un requisito con una estabilidad baja es bastante sensible a cambios.

4.2.1 Lista de nombres de requisitos

Para una mayor comprensión de los requisitos, se adjuntan unas tablas con los nombres de todos los requisitos y un acceso rápido a los mismos.

4.2.1.1 Nombres de requisitos de capacidad

Identificador	Nombre
RUC-01	**Gestión de reglas de juego**
RUC-02	**Gestión de escenarios**
RUC-03	**Consulta**
RUC-04	**Flexibilidad**
RUC-05	**Ensamblado (I)**
RUC-06	**Ensamblado (II)**
RUC-07	**Generación de XML**

Tabla 10. Nombres de reqs. de capacidad

4.2.1.2 Nombres de requisitos de restricción

Identificador	Nombre
RUR-01	Accesibilidad
RUR-02	Aplicación web
RUR-03	Compatibilidad con navegadores
RUR-04	Idioma
RUR-05	Usabilidad
RUR-06	Introducción de información
RUR-07	Diseño rápido
RUR-08	Almacenamiento persistente

Tabla 11. Nombres de reqs. de restricción

4.2.2 Requisitos de capacidad

Identificador	RUC-01		
Nombre	Gestión de reglas de juego		
Descripción	La herramienta permitirá una gestión completa de las reglas de juego: • Creación de elementos de reglas. • Edición de elementos de reglas. • Borrado de elementos de reglas. • Agregación de elementos a conjuntos de reglas. • Creación, edición y borrado de conjuntos de reglas.		
Fuente	Cliente	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 12. Requisito de capacidad RUC-01

Identificador	RUC-02		
Nombre	Gestión de escenarios		
Descripción	La herramienta permitirá una gestión completa de escenarios: • Creación de elementos de escenario. • Edición de elementos de escenario. • Borrado de elementos de escenario. • Agregación de elementos a escenarios. • Creación, edición y borrado de escenarios.		
Fuente	Cliente	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 13. Requisito de capacidad RUC-02

Identificador	RUC-03		
Nombre	Consulta		
Descripción	El usuario podrá consultar la información creada.		
Fuente	Cliente	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 14. Requisito de capacidad RUC-03

Identificador	RUC-04		
Nombre	Flexibilidad		
Descripción	Si surgen nuevos elementos de diseño a considerar, la herramienta será fácilmente ampliable.		
Fuente	Cliente	Prioridad	Alta
Necesidad	Deseable	Estabilidad	Alta

Tabla 15. Requisito de capacidad RUC-04

Identificador	RUC-05		
Nombre	Ensamblado (I)		
Descripción	La herramienta permitirá el ensamblado de conjuntos de reglas y escenarios seleccionados por el usuario para dar lugar al diseño final del videojuego.		
Fuente	Cliente	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 16. Requisito de capacidad RUC-05

Identificador	RUC-06		
Nombre	Ensamblado (II)		
Descripción	La herramienta ofrecerá la posibilidad de obtener una combinación de videojuegos a nivel de mini-juegos a partir de la combinación de diferentes storytellings creados por el usuario.		
Fuente	Cliente	Prioridad	Baja
Necesidad	Deseable	Estabilidad	Alta

Tabla 17. Requisito de capacidad RUC-06

Identificador	RUC-07		
Nombre	Generación de XML		
Descripción	La herramienta generará documentos XML con el diseño del videojuego a partir de toda la información aportada por el usuario		
Fuente	Cliente	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 18. Requisito de capacidad RUC-07

4.2.3 Requisitos de restricción

Identificador	RUR-01		
Nombre	Accesibilidad		
Descripción	La herramienta de autoría será accesible desde ordenador y desde *tablets*.		
Fuente	Cliente	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 19. Requisito de restricción RUR-01

Identificador	RUR-02		
Nombre	Aplicación web		
Descripción	La herramienta de autoría será una aplicación web.		
Fuente	Cliente	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 20. Requisito de restricción RUR-02

Identificador	RUR-03		
Nombre	Compatibilidad con navegadores		
Descripción	La herramienta será compatible con el mayor número de navegadores posible.		
Fuente	Cliente	Prioridad	Alta
Necesidad	Deseable	Estabilidad	Alta

Tabla 21. Requisito de restricción RUR-03

Identificador	RUR-04		
Nombre	Idioma		
Descripción	El idioma de la herramienta será el inglés.		
Fuente	Cliente	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 22. Requisito de restricción RUR-04

Identificador	RUR-05		
Nombre	Usabilidad		
Descripción	La herramienta de autoría será sencilla de utilizar. El usuario principal será un educador con un perfil tecnológico posiblemente bajo.		
Fuente	Cliente	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 23. Requisito de restricción RUR-05

Identificador	RUR-06		
Nombre	Introducción de información		
Descripción	La introducción de la información se hará a través de campos de formularios.		
Fuente	Cliente	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 24. Requisito de restricción RUR-06

Identificador	RUR-07		
Nombre	Diseño rápido		
Descripción	La herramienta permitirá diseñar reglas y escenarios de forma rápida a través de: • Creación de plantillas de reglas o escenarios que podrán ser utilizadas en diferentes videojuegos. • Atajos a la hora de introducir la información.		
Fuente	Cliente	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 25. Requisito de restricción RUR-07

Identificador	RUR-08		
Nombre	Almacenamiento persistente		
Descripción	Toda la información introducida por el usuario se guardará de manera persistente en la herramienta.		
Fuente	Cliente	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 26. Requisito de restricción RUR-08

4.3 Catálogo de requisitos de software

Los requisitos de software son una especificación técnica de los requisitos de usuario cuyo principal interesado en ellos es el desarrollador del sistema. Existen dos tipos principales de requisitos de software:

- **Requisitos funcionales**. Especifican la funcionalidad o servicios que el sistema debe proporcionar.

- **Requisitos no funcionales**. Imponen restricciones en el sistema o en el proceso de desarrollo del mismo. De esta última clasificación derivan los siguientes tipos de requisitos:

    - **Requisitos de operación**. Indican cómo el sistema va a ejecutar las acciones que proveerán el correcto funcionamiento del mismo.

    - **Requisitos de interfaz**. Describen el formato con el que el sistema se comunica con su entorno.

- **Requisitos de rendimiento**. Especifican restricciones temporales que el sistema ha de cumplir.
- **Requisitos de recursos**. Definen los límites posibles de los recursos físicos que va a utilizar el sistema.
- **Requisitos de seguridad**. Detallan la seguridad del sistema frente a amenazas externas que pongan en entredicho la confidencialidad, la integridad y la disponibilidad.
- **Requisitos de usabilidad**. Describen la facilidad de uso del sistema por parte del usuario al que va destinado.
- **Requisitos de comprobación**. Indican de qué modo el sistema verifica que el formato de la información introducida es el correcto.

Del mismo modo que con los requisitos de usuario, los de software van a ser presentados en un formato tabular similar. La única diferencia radica en el identificador, que en el caso de los requisitos de software va a respetar la siguiente nomenclatura: **RS(F/NF)-YY**. **RS** hace referencia a un requisito de software. **F** indica que el requisito es funcional y **NF** que es no funcional. **YY** es un valor numérico que oscila entre 01 y 99.

4.3.1 Lista de nombres de requisitos

Para una mayor comprensión de los requisitos, se adjuntan unas tablas con los nombres de todos los requisitos y un acceso rápido a los mismos.

4.3.1.1 Nombres de requisitos funcionales

Identificador	Nombre	Identificador	Nombre
RSF-01	Niveles de elementos de reglas	RSF-19	Creación de elementos de contexto
RSF-02	Creación de entidades	RSF-20	Creación de servicios
RSF-03	Creación de eventos	RSF-21	Creación de interacciones
RSF-04	Creación de objetivos	RSF-22	Edición de elementos de escenario
RSF-05	Creación de feedbacks	RSF-23	Borrado de elementos de escenario
RSF-06	Creación de storytellings	RSF-24	Consulta de elementos de escenario

RSF-07	Edición de elementos de reglas	RSF-25	Creación, borrado y edición de subelementos de escenario
RSF-08	Borrado de elementos de reglas	RSF-26	Consulta de subelementos de escenario
RSF-09	Consulta de elementos de reglas	RSF-27	Creación de escenarios
RSF-10	Creación, borrado y edición de subelementos de reglas	RSF-28	Edición de escenarios
RSF-11	Consulta de subelementos de reglas	RSF-29	Borrado de escenarios
RSF-12	Creación de reglas	RSF-30	Consulta de escenarios
RSF-13	Edición de reglas	RSF-31	Creación de matches
RSF-14	Borrado de reglas	RSF-32	Borrado de matches
RSF-15	Consulta de reglas	RSF-33	Carga de matches
RSF-16	Generación de diseño con storytellings	RSF-34	Unión de reglas y escenario
RSF-17	Niveles de elementos de escenario	RSF-35	Generación del diseño final
RSF-18	Creación de entidades de escenario		

Tabla 27. Nombres de reqs. Funcionales

4.3.1.2 Nombres de requisitos no funcionales

Identificador	Nombre	Identificador	Nombre
RSNF-01	Compatibilidad con navegadores	RSNF-13	Carga de páginas
RSNF-02	Sobreescritura de elementos	RSNF-14	Operaciones del sistema
RSNF-03	Distinción de subelementos de entidades de reglas	RSNF-15	Soporte *tablet*
RSNF-04	Sistema gestor de BBDD	RSNF-16	Wi-Fi

RSNF-05	Compatibilidad con HTML5 y CSS3	RSNF-17	Servidor del sistema
RSNF-06	Agregación de subelementos	RSNF-18	Prevención de inyecciones SQL
RSNF-07	Agregación de elementos	RSNF-19	Prevención de ataques XSS
RSNF-08	Interfaz adaptada	RSNF-20	Instrucciones
RSNF-09	Idioma del sistema	RSNF-21	Campos de formulario obligatorios
RSNF-10	Uso de formularios	RSNF-22	Campos de formulario incorrectos
RSNF-11	Formularios dinámicos	RSNF-23	Formato de la información
RSNF-12	Campos obligatorios	RSNF-24	Verificación de campos obligatorios

Tabla 28. Nombres de reqs. no funcionales

4.3.2 Requisitos funcionales

4.3.2.1 Reglas

Identificador	**RSF-01**		
Nombre	**Niveles de elementos de reglas**		
Descripción	**Existirán tres niveles de elementos de reglas, que son:** • **Nivel 1: Entidades y eventos.** • **Nivel 2: Objetivos.** • **Nivel 3: Feedback y Storytelling.**		
Fuente	**Analista**	**Prioridad**	**Alta**
Necesidad	**Esencial**	**Estabilidad**	**Alta**

Tabla 29. Requisito funcional RSF-01

Identificador	RSF-02		
Nombre	Creación de entidades		
Descripción	El usuario podrá crear entidades y almacenarlas de manera persistente aportando la siguiente información: • Nombre. • Tipo. • Descripción. • Subelementos (estados/comportamientos, atributos y acciones).		
Fuente	Analista	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 30. Requisito funcional RSF-02

Identificador	RSF-03		
Nombre	Creación de eventos		
Descripción	El usuario podrá crear eventos y almacenarlos de manera persistente facilitando la siguiente información: • Nombre. • Descripción. • Subelementos (condición y consecuencias).		
Fuente	Analista	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 31. Requisito funcional RSF-03

Identificador	RSF-04
Nombre	Creación de objetivos
Descripción	El usuario será capaz de crear objetivos y almacenarlos de manera persistente aportando los siguientes datos: • Nombre. • Tipo. • Descripción. • Evento. • Threshold.

Fuente	Analista	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 32. Requisito funcional RSF-04

Identificador	RSF-05		
Nombre	Creación de feedbacks		
Descripción	El usuario podrá crear feedbacks y guardarlos persistentemente facilitando los siguientes datos: • Nombre. • Tipo. • Evento. • Valor.		
Fuente	Analista	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 33. Requisito funcional RSF-05

Identificador	RSF-06		
Nombre	Creación de storytellings		
Descripción	El usuario podrá crear storytellings y guardarlos de manera persistente aportando la siguiente información: • Nombre. • Subelementos (episodios y subelementos propios).		
Fuente	Analista	Prioridad	Baja
Necesidad	Deseable	Estabilidad	Alta

Tabla 34. Requisito funcional RSF-06

Identificador	RSF-07
Nombre	Edición de elementos de reglas
Descripción	El usuario tendrá capacidad para editar cualquier tipo de elemento de reglas y salvar las modificaciones de manera permanente.

Fuente	Analista	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 35. Requisito funcional RSF-07

Identificador	RSF-08		
Nombre	Borrado de elementos de reglas		
Descripción	El usuario podrá borrar permanentemente cualquier tipo de elemento de reglas guardado.		
Fuente	Analista	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 36. Requisito funcional RSF-08

Identificador	RSF-09		
Nombre	Consulta de elementos de reglas		
Descripción	El usuario podrá consultar la información relativa de cualquier tipo de elemento de reglas guardado en el sistema.		
Fuente	Analista	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 37. Requisito funcional RSF-09

Identificador	RSF-10		
Nombre	Creación, borrado y edición de subelementos de reglas		
Descripción	El usuario podrá crear, borrar y editar cualquier tipo de subelemento de reglas.		
Fuente	Analista	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 38. Requisito funcional RSF-10

Identificador	RSF-11		
Nombre	Consulta de subelementos de reglas		
Descripción	El usuario podrá consultar la información relativa de cualquier tipo de subelemento de reglas guardado en el sistema.		
Fuente	Analista	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 39. Requisito funcional RSF-11

Identificador	RSF-12		
Nombre	Creación de reglas		
Descripción	El usuario será capaz de crear conjuntos de reglas y guardarlos de manera persistente aportando la siguiente información: • Nombre. • Descripción. • Entidades. • Eventos. • Objetivos. • Feedbacks.		
Fuente	Analista	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 40. Requisito funcional RSF-12

Identificador	RSF-13		
Nombre	Edición de reglas		
Descripción	El usuario tendrá capacidad para editar cualquier conjunto de reglas y salvar las modificaciones de manera permanente.		
Fuente	Analista	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 41. Requisito funcional RSF-13

Identificador	RSF-14		
Nombre	Borrado de reglas		
Descripción	El usuario podrá borrar persistentemente cualquier conjunto de reglas guardado.		
Fuente	Analista	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 42. Requisito funcional RSF-14

Identificador	RSF-15		
Nombre	Consulta de reglas		
Descripción	El usuario podrá consultar la información relativa de cualquier conjunto de reglas guardado en el sistema.		
Fuente	Analista	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 43. Requisito funcional RSF-15

Identificador	RSF-16		
Nombre	Generación de diseño con storytellings		
Descripción	El sistema podrá generar un diseño final de videojuego a partir de la unión de varios storytellings almacenados.		
Fuente	Analista	Prioridad	Baja
Necesidad	Deseable	Estabilidad	Alta

Tabla 44. Requisito funcional RSF-16

4.3.2.2 Escenarios

Identificador	RSF-17		
Nombre	Niveles de elementos de escenario		
Descripción	Existirán tres niveles de elementos de escenario, que son: • Nivel 1: Entidades de escenario y elementos de contexto. • Nivel 2: Servicios. • Nivel 3: Interacciones.		
Fuente	Analista	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 45. Requisito funcional RSF-17

Identificador	RSF-18		
Nombre	Creación de entidades de escenario		
Descripción	El usuario podrá crear entidades de escenario y almacenarlas de manera persistente aportando la siguiente información: • Nombre. • Descripción. • Subelementos (nombres de ficheros de modelos gráficos). • Carácter principal. • Nombre de fichero por defecto.		
Fuente	Analista	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 46. Requisito funcional RSF-18

Identificador	RSF-19
Nombre	Creación de elementos de contexto
Descripción	El usuario podrá crear elementos de contexto y almacenarlos de manera persistente facilitando la siguiente información: • Nombre. • Valor.

Fuente	Analista	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 47. Requisito funcional RSF-19

Identificador	RSF-20		
Nombre	Creación de servicios		
Descripción	El usuario será capaz de crear servicios y almacenarlos de manera persistente aportado los siguientes datos: • Nombre. • Valor. • Posición. • Tamaño.		
Fuente	Analista	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 48. Requisito funcional RSF-20

Identificador	RSF-21		
Nombre	Creación de interacciones		
Descripción	El usuario podrá crear interacciones y guardarlos persistentemente facilitando los siguientes datos: • Dispositivo. • Control. • Control alternativo.		
Fuente	Analista	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 49. Requisito funcional RSF-21

Identificador	RSF-22
Nombre	Edición de elementos de escenario
Descripción	El usuario tendrá capacidad para editar cualquier tipo de elemento de escenario y salvar las modificaciones de manera permanente.

Fuente	**Analista**	**Prioridad**	**Alta**
Necesidad	**Esencial**	**Estabilidad**	**Alta**

Tabla 50. Requisito funcional RSF-22

Identificador	**RSF-23**		
Nombre	**Borrado de elementos de escenario**		
Descripción	**El usuario podrá borrar permanentemente cualquier tipo de elemento de escenario guardado.**		
Fuente	**Analista**	**Prioridad**	**Alta**
Necesidad	**Esencial**	**Estabilidad**	**Alta**

Tabla 51. Requisito funcional RSF-23

Identificador	**RSF-24**		
Nombre	**Consulta de elementos de escenario**		
Descripción	**El usuario podrá consultar la información relativa de cualquier tipo de elemento de escenario guardado en el sistema.**		
Fuente	**Analista**	**Prioridad**	**Alta**
Necesidad	**Esencial**	**Estabilidad**	**Alta**

Tabla 52. Requisito funcional RSF-24

Identificador	**RSF-25**		
Nombre	**Creación, borrado y edición de subelementos de escenario**		
Descripción	**El usuario podrá crear, borrar y editar cualquier tipo de subelemento de escenario.**		
Fuente	**Analista**	**Prioridad**	**Alta**
Necesidad	**Esencial**	**Estabilidad**	**Alta**

Tabla 53. Requisito funcional RSF-25

Identificador	RSF-26		
Nombre	Consulta de subelementos de escenario		
Descripción	El usuario podrá consultar la información relativa de cualquier tipo de subelemento de escenario guardado en el sistema.		
Fuente	Analista	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 54. Requisito funcional RSF-26

Identificador	RSF-27		
Nombre	Creación de escenarios		
Descripción	El usuario será capaz de crear escenarios y guardarlos de manera persistente aportando la siguiente información: • Nombre. • Descripción. • Entidades de escenario. • Elementos de contexto. • Servicios. • Interacciones.		
Fuente	Analista	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 55. Requisito funcional RSF-27

Identificador	RSF-28		
Nombre	Edición de escenarios		
Descripción	El usuario tendrá capacidad para editar cualquier escenario y salvar las modificaciones de manera permanente.		
Fuente	Analista	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 56. Requisito funcional RSF-28

Identificador	RSF-29		
Nombre	Borrado de escenarios		
Descripción	El usuario podrá borrar persistentemente cualquier escenario guardado.		
Fuente	Analista	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 57. Requisito funcional RSF-29

Identificador	RSF-30		
Nombre	Consulta de escenarios		
Descripción	El usuario podrá consultar la información relativa de cualquier escenario guardado en el sistema.		
Fuente	Analista	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 58. Requisito funcional RSF-30

4.3.2.3 Unión de reglas y escenario y generación del diseño

Identificador	RSF-31		
Nombre	Creación de matches		
Descripción	El usuario podrá crear matches y almacenarlos de manera persistente aportando la siguiente información: • Nombre. • Conjunto de reglas. • Escenario.		
Fuente	Analista	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 59. Requisito funcional RSF-31

Identificador	RSF-32		
Nombre	Borrado de matches		
Descripción	El usuario será capaz de borrar cualquier match guardado en el sistema.		
Fuente	Analista	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 60. Requisito funcional RSF-32

Identificador	RSF-33		
Nombre	Carga de matches		
Descripción	El usuario podrá cargar un match almacenado y visualizar su contenido.		
Fuente	Analista	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 61. Requisito funcional RSF-33

Identificador	RSF-34		
Nombre	Unión de reglas y escenario		
Descripción	El sistema podrá unir un conjunto de reglas y un escenario para dar lugar a un match.		
Fuente	Analista	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 62. Requisito funcional RSF-34

Identificador	RSF-35		
Nombre	Generación del diseño final		
Descripción	El sistema generará el diseño final a partir de la información contenida en un match.		
Fuente	Analista	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 63. Requisito funcional RSF-35

4.3.3 Requisitos no funcionales

4.3.3.1 Requisitos de operación

Identificador	RSNF-01		
Nombre	Compatibilidad con navegadores		
Descripción	El sistema será compatible con los navegadores web principales del mercado: Google Chrome, Firefox, Internet Explorer y Opera.		
Fuente	Analista	Prioridad	Alta
Necesidad	Deseable	Estabilidad	Alta

Tabla 64. Requisito no funcional RSNF-01

Identificador	RSNF-02		
Nombre	Sobreescritura de elementos		
Descripción	El sistema permitirá la sobreescritura de cualquier tipo de elemento aunque no se haya modificado ningún atributo suyo guardado en base de datos.		
Fuente	Analista	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 65. Requisito no funcional RSNF-02

Identificador	RSNF-03		
Nombre	Distinción de subelementos de entidades de reglas		
Descripción	El sistema hará distinción entre subelementos de entidades de reglas generales y subelementos de entidades de reglas asociados a alguna en concreto, diferenciando de este modo las operaciones de guardado y borrado entre ellos.		
Fuente	Analista	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 66. Requisito no funcional RSNF-03

Identificador	RSNF-04		
Nombre	Sistema gestor de BBDD		
Descripción	Se utilizará el sistema gestor de bases de datos relacionales MySQL para la aplicación de la persistencia de información.		
Fuente	Analista	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 67. Requisito no funcional RSNF-04

Identificador	RSNF-05		
Nombre	Compatibilidad con HTML5 y CSS3		
Descripción	El sistema será compatible con HTML5 y CSS3.		
Fuente	Analista	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 68. Requisito no funcional RSNF-05

Identificador	RSNF-06		
Nombre	Agregación de subelementos		
Descripción	El sistema permitirá agregar una sola instancia de un subelemento de reglas/escenario en un elemento de reglas/escenario.		
Fuente	Analista	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 69. Requisito no funcional RSNF-06

Identificador	RSNF-07
Nombre	Agregación de elementos
Descripción	El sistema permitirá agregar una sola instancia de un elemento de reglas/escenario en un conjunto de reglas/escenario.

Fuente	Analista	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 70. Requisito no funcional RSNF-07

4.3.3.2 Requisitos de interfaz

Identificador	RSNF-08		
Nombre	Interfaz adaptada		
Descripción	El sistema presentará una interfaz adaptada tanto a la resolución de ordenadores (portátiles o sobremesa) como a la de *tablets*.		
Fuente	Analista	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 71. Requisito no funcional RSNF-08

Identificador	RSNF-09		
Nombre	Idioma del sistema		
Descripción	El sistema presentará una interfaz escrita totalmente en inglés.		
Fuente	Analista	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 72. Requisito no funcional RSNF-09

Identificador	RSNF-10		
Nombre	Uso de formularios		
Descripción	El sistema ofrecerá formularios web para introducir la información requerida para la definición de diseños.		
Fuente	Analista	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 73. Requisito no funcional RSNF-10

Identificador	RSNF-11		
Nombre	Formularios dinámicos		
Descripción	Los formularios serán dinámicos, es decir, se adaptarán a la información que aporte el usuario.		
Fuente	Analista	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 74. Requisito no funcional RSNF-11

Identificador	RSNF-12		
Nombre	Campos obligatorios		
Descripción	El sistema indicará al usuario qué campos de formulario son obligatorios.		
Fuente	Analista	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 75. Requisito no funcional RSNF-12

4.3.3.3 Requisitos de rendimiento

Identificador	RSNF-13		
Nombre	Carga de páginas		
Descripción	El sistema tardará como máximo cinco segundos en cargar una página.		
Fuente	Analista	Prioridad	Media
Necesidad	Deseable	Estabilidad	Alta

Tabla 76. Requisito no funcional RSNF-13

Identificador	RSNF-14		
Nombre	Operaciones del sistema		
Descripción	Las operaciones del sistema conllevarán una espera de cómo máximo 4 segundos.		
Fuente	Analista	Prioridad	Media
Necesidad	Deseable	Estabilidad	Alta

Tabla 77. Requisito no funcional RSNF-14

4.3.3.4 Requisitos de recursos

Identificador	RSNF-15		
Nombre	Soporte *tablet*		
Descripción	El sistema necesitará un soporte *tablet* de gama media/alta para ser ejecutado correctamente.		
Fuente	Analista	Prioridad	Media
Necesidad	Deseable	Estabilidad	Alta

Tabla 78. Requisito no funcional RSNF-15

Identificador	RSNF-16		
Nombre	Wi-Fi		
Descripción	Se requerirá una conexión *Wi-Fi* estable para evitar el malfuncionamiento del sistema.		
Fuente	Analista	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 79. Requisito no funcional RSNF-16

Identificador	RSNF-17		
Nombre	Servidor del sistema		
Descripción	El servidor del sistema deberá permanecer en activo para llevar a cabo todas las operaciones.		
Fuente	Analista	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 80. Requisito no funcional RSNF-17

4.3.3.5 Requisitos de seguridad

Identificador	RSNF-18		
Nombre	Prevención de inyecciones SQL		
Descripción	Las consultas a la base de datos se construirán de acuerdo a prevenir posibles ataques de inyección SQL.		
Fuente	Analista	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 81. Requisito no funcional RSNF-18

Identificador	RSNF-19		
Nombre	Prevención de ataques XSS		
Descripción	El sistema comprobará el formato de los datos introducidos con vistas a prevenir posibles ataques XSS (Cross-Site Scripting).		
Fuente	Analista	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 82. Requisito no funcional RSNF-19

4.3.3.6 Requisitos de usabilidad

Identificador	RSNF-20		
Nombre	Instrucciones		
Descripción	El sistema ofrecerá instrucciones en cada sección con el objetivo de orientar al usuario.		
Fuente	Analista	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 83. Requisito no funcional RSNF-20

Identificador	RSNF-21		
Nombre	Campos de formulario obligatorios		
Descripción	El sistema indicará al usuario los campos de formulario obligatorios que ha de rellenar en caso de que se le olvide alguno en el momento de enviar el formulario.		
Fuente	Analista	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 84. Requisito no funcional RSNF-21

Identificador	RSNF-22		
Nombre	Campos de formulario incorrectos		
Descripción	El sistema indicará al usuario los campos de formulario que haya introducido incorrectamente y sugerencias para corregirlos en el momento de enviar el formulario.		
Fuente	Analista	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 85. Requisito no funcional RSNF-22

4.3.3.7 Requisitos de comprobación

Identificador	RSNF-23		
Nombre	Formato de la información		
Descripción	El sistema verificará el formato de la información introducida en cada campo de formulario en busca de inconsistencias.		
Fuente	Analista	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 86. Requisito no funcional RSNF-23

Identificador	RSNF-24		
Nombre	Verificación de campos obligatorios		
Descripción	El sistema comprobará que se introduzcan todos los campos obligatorios de cada formulario.		
Fuente	Analista	Prioridad	Alta
Necesidad	Esencial	Estabilidad	Alta

Tabla 87. Requisito no funcional RSNF-24

4.4 Casos de uso

Los casos de uso son descripciones de las interacciones entre el usuario y el sistema desde el punto de vista del usuario. Su uso permite concretar mejor el objetivo de cada uno de los requisitos definidos.

Existen dos formas de representar los casos de uso: a través de un diagrama de casos de uso, en el que se muestran los actores implicados y las diferentes interacciones con el sistema, o por medio de notación escrita, con la que se indica al menos el objetivo del caso de uso, los actores implicados y los cursos básicos y alternativos de acciones asociadas al mencionado caso de uso.

En primer lugar, se va a presentar el diagrama de casos de uso correspondiente a la herramienta de autoría:

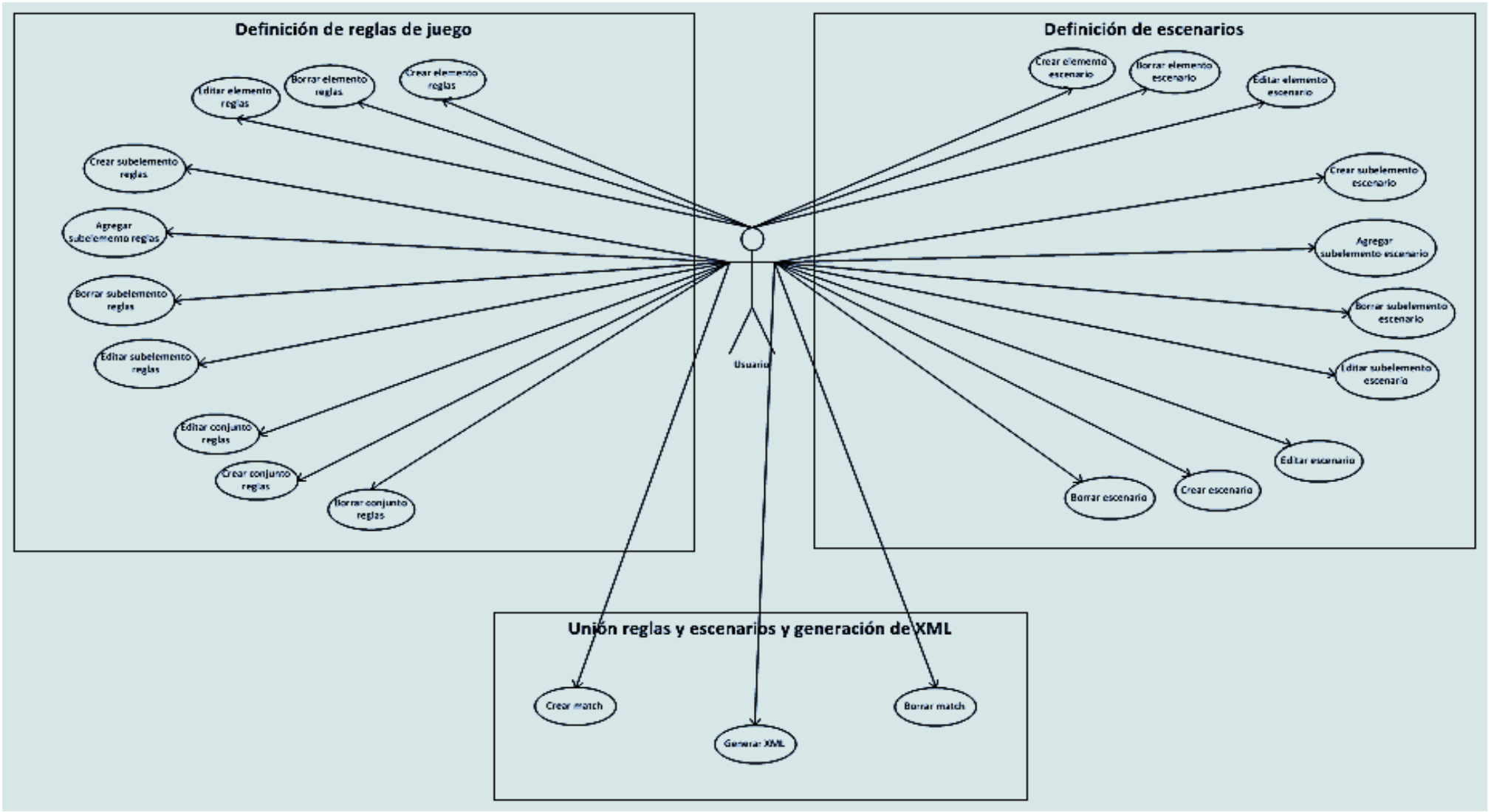


Ilustración 17. Diagrama de casos de uso

Se entiende por subelemento de reglas/escenario a todo elemento que forma parte de la estructura final de un elemento de reglas/escenario. Un elemento de reglas/escenario es aquel que forma parte de un conjunto de reglas (entidad, objetivo, feedback, etc.) o de un escenario (servicio, interacción, etc.). Un match es la unión de un conjunto de reglas de juego y de un escenario.

Del mismo modo que con los requisitos, la descripción de cada caso de uso se va a ofrecer a través de tablas con unos determinados campos. El formato de tabla y la explicación de los campos se presentan de inmediato:

Identificador	
Nombre	
Descripción	
Actor	
Precondiciones	
Flujo básico	
Flujo alternativo	
Postcondiciones	

Tabla 88. Tabla casos de uso modelo

- **Identificador**. Permite registrar de manera única un caso de uso de acuerdo con una determinada nomenclatura: **CU-XX**. **CU** hace referencia a un caso de uso. **XX** es un valor numérico que oscila entre 01 y 99.

- **Nombre**. Descripción breve del caso de uso.

- **Descripción**. Explicación detallada y clara del objetivo del caso de uso.

- **Actor**. Persona o sistema que interactúa con la herramienta.

- **Precondiciones**. Condiciones previas que han de cumplirse antes de ejecutar el caso de uso.

- **Flujo básico**. Secuencia de interacciones entre actor y sistema que culmina en el desarrollo correcto del caso de uso.

- **Flujo alternativo**. Secuencia de interacciones que se produce por alguna circunstancia especial y que constituye una alternativa al flujo básico.

- **Postcondiciones**. Condiciones que se dan al finalizar el caso de uso.

4.4.1 Lista de casos de uso

Identificador	**CU-01**
Nombre	**Crear subelemento reglas**
Descripción	**El usuario crea un subelemento de reglas rellenando los campos necesarios del formulario asociado.**
Actor	**Usuario**
Precondiciones	**La herramienta muestra al usuario la página inicial de la sección de definición de reglas de juego.**
Flujo básico	**1. El usuario escoge un elemento de reglas del submenú que hay presente en la página.** **2. El sistema muestra el formulario de creación del elemento de reglas.** **3. El usuario pulsa el botón de crear subelemento de reglas.** **4. El sistema ofrece una ventana modal al usuario con el formulario de creación del subelemento.** **5. El usuario rellena todos los campos necesarios y pulsa el botón de guardar.** **6. El sistema almacena en la base de datos el subelemento creado por el usuario.** **7. El sistema cierra la ventana modal y devuelve la navegación a la página con el formulario de creación del elemento de reglas.**
Flujo alternativo	**5. a) El usuario pulsa el botón de cierre de la ventana.** **1. El sistema cierra la ventana modal con el formulario.** **2. Se muestra de nuevo el formulario de creación del elemento de reglas.**
Flujo alternativo	**5. b) El usuario no rellena todos los campos obligatorios y pulsa el botón de guardar.** **1. El sistema le indica al usuario los campos que son obligatorios y no ejecuta ninguna acción de guardado.** **2. Se retoma el flujo básico en el inicio del paso 5.**
Flujo alternativo	**5. c) El usuario rellena de manera incorrecta los campos del formulario y pulsa el botón de guardar.** **1. El sistema le indica al usuario los campos erróneos y proporciona una ayuda para su corrección.** **2. Se retoma el flujo básico en el inicio del paso 5.**
Postcondiciones	**La herramienta carga la página con el formulario de creación del elemento de reglas correspondiente.**

Tabla 89. Caso de uso CU-01

Identificador	**CU-02**
Nombre	**Borrar subelemento reglas**
Descripción	**El usuario borra un subelemento de reglas, desapareciendo éste de la base de datos.**
Actor	**Usuario**
Precondiciones	**La herramienta muestra al usuario la página inicial de la sección de definición de reglas de juego.**
Flujo básico	**1. El usuario escoge un elemento de reglas del submenú que hay presente en la página.** **2. El sistema muestra el formulario de creación del elemento de reglas.** **3. El usuario pulsa el botón de agregar subelemento de reglas.** **4. El sistema ofrece una ventana modal al usuario con la lista de subelementos de reglas guardados y un pequeño campo de búsqueda.** **5. El usuario borra el subelemento que desee pulsando el botón de borrar correspondiente.** **6. El sistema elimina de la base de datos el subelemento elegido.** **7. El sistema cierra la ventana modal y devuelve la navegación a la página con el formulario de creación del elemento de reglas.**
Flujo alternativo	**5. a) El usuario pulsa el botón de cierre de la ventana.** **1. El sistema cierra la ventana modal con el formulario.** **2. Se muestra de nuevo el formulario de creación del elemento de reglas.**
Flujo alternativo	**5. b) El usuario introduce el nombre de un subelemento en el campo de búsqueda y pulsa el botón de buscar.** **1. El sistema muestra una nueva lista con los resultados de la búsqueda.** **2. El usuario borra el subelemento que desee pulsando el botón de borrar correspondiente.** **3. Se retoma el flujo básico en el paso 6.**
Postcondiciones	**La herramienta carga la página con el formulario de creación del elemento de reglas correspondiente.**

Tabla 90. Caso de uso CU-02

Identificador	**CU-03**
Nombre	**Editar subelemento reglas**
Descripción	**El usuario edita un subelemento de reglas, modificando los campos que crea convenientes.**
Actor	**Usuario**
Precondiciones	**La herramienta muestra al usuario la página inicial de la sección de definición de reglas de juego.**
Flujo básico	**1. El usuario escoge un elemento de reglas del submenú que hay presente en la página.** **2. El sistema muestra el formulario de creación del elemento de reglas.** **3. El usuario pulsa el botón de agregar subelemento de reglas.** **4. El sistema ofrece una ventana modal al usuario con la lista de subelementos de reglas guardados y un pequeño campo de búsqueda.** **5. El usuario pulsa el botón de editar correspondiente al subelemento elegido.** **6. El sistema ofrece una ventana modal al usuario con el formulario de modificación del subelemento.** **7. El usuario rellena o modifica los campos que él decida y pulsa el botón de guardar.** **8. El sistema almacena en la base de datos el subelemento modificado por el usuario.** **9. El sistema cierra la ventana modal y devuelve la navegación a la página con el formulario de creación del elemento de reglas.**
Flujo alternativo	**5. a) El usuario pulsa el botón de cierre de la ventana.** **1. El sistema cierra la ventana modal con el formulario.** **2. Se muestra de nuevo el formulario de creación del elemento de reglas.**
Flujo alternativo	**5. b) El usuario introduce el nombre de un subelemento en el campo de búsqueda y pulsa el botón de buscar.** **1. El sistema muestra una nueva lista con los resultados de la búsqueda.** **2. El usuario pulsa el botón de editar correspondiente al subelemento elegido.** **3. Se retoma el flujo básico en el paso 6.**
Flujo alternativo	**7. a) El usuario pulsa el botón de cierre de la ventana.** **1. El sistema cierra la ventana modal con el formulario.** **2. Se muestra de nuevo el formulario de creación del elemento de reglas.**
Flujo alternativo	**7. b) El usuario no rellena todos los campos obligatorios y pulsa el botón de guardar.** **1. El sistema le indica al usuario los campos que son**

	obligatorios y no ejecuta ninguna acción de guardado. **2. Se retoma el flujo básico en el inicio del paso 7.**
Flujo alternativo	**7. c) El usuario rellena de manera incorrecta los campos del formulario y pulsa el botón de guardar.** **1. El sistema le indica al usuario los campos erróneos y proporciona una ayuda para su corrección.** **2. Se retoma el flujo básico en el inicio del paso 7.**
Postcondiciones	**La herramienta carga la página con el formulario de creación del elemento de reglas correspondiente.**

Tabla 91. Caso de uso CU-03

Identificador	**CU-04**
Nombre	**Agregar subelemento reglas**
Descripción	**El usuario agrega un subelemento de reglas al correspondiente elemento de reglas al que pertenece.**
Actor	**Usuario**
Precondiciones	**La herramienta muestra al usuario la página inicial de la sección de definición de reglas de juego.**
Flujo básico	**1. El usuario escoge un elemento de reglas del submenú que hay presente en la página.** **2. El sistema muestra el formulario de creación del elemento de reglas.** **3. El usuario pulsa el botón de agregar subelemento de reglas.** **4. El sistema ofrece una ventana modal al usuario con la lista de subelementos de reglas guardados y un pequeño campo de búsqueda.** **5. El usuario selecciona el subelemento de reglas que desee y pulsa el botón de agregar.** **6. El sistema cierra la ventana modal y devuelve la navegación a la página con el formulario de creación del elemento de reglas.**
Flujo alternativo	**5. a) El usuario pulsa el botón de cierre de la ventana.** **1. El sistema cierra la ventana modal con el formulario** **2. Se muestra de nuevo el formulario de creación del elemento de reglas.**
Flujo alternativo	**5. b) El usuario introduce el nombre de un subelemento en el campo de búsqueda y pulsa el botón de buscar.** **1. El sistema muestra una nueva lista con los resultados de la búsqueda.** **2. El usuario selecciona el subelemento de reglas que desee y pulsa el botón de agregar.** **3. Se retoma el flujo básico en el paso 6.**

Postcondiciones	**La herramienta carga la página con el subelemento incluido en el formulario de creación del elemento de reglas correspondiente.**

Tabla 92. Caso de uso CU-04

Identificador	**CU-05**
Nombre	**Crear elemento reglas**
Descripción	**El usuario crea un elemento de reglas rellenando los campos necesarios del formulario asociado.**
Actor	**Usuario**
Precondiciones	**La herramienta muestra al usuario la página inicial de la sección de definición de reglas de juego.**
Flujo básico	**1. El usuario escoge un elemento de reglas del submenú que hay presente en la página.** **2. El sistema muestra el formulario de creación del elemento de reglas.** **3. El usuario rellena todos los campos necesarios y ejecuta las acciones de agregar subelementos en caso de que los haya.** **4. El usuario pulsa el botón de guardar.** **5. El sistema almacena en la base de datos el elemento creado por el usuario.** **6. El sistema muestra un mensaje notificando el correcto guardado del elemento de reglas.** **7. El sistema recarga la página.**
Flujo alternativo	**4. a) El usuario no rellena todos los campos obligatorios y pulsa el botón de guardar.** **1. El sistema le indica al usuario los campos que son obligatorios y no ejecuta ninguna acción de guardado.** **2. Se retoma el flujo básico en el inicio del paso 3.**
Flujo alternativo	**4. b) El usuario rellena de manera incorrecta los campos del formulario y pulsa el botón de guardar.** **1. El sistema le indica al usuario los campos erróneos y proporciona una ayuda para su corrección.** **2. Se retoma el flujo básico en el inicio del paso 3.**
Postcondiciones	**La herramienta muestra la página con el formulario de creación del elemento de reglas vacío.**

Tabla 93. Caso de uso CU-05

Identificador	CU-06
Nombre	Borrar elemento reglas
Descripción	El usuario borra un elemento de reglas, desapareciendo éste de la base de datos junto con todos sus subelementos asociados.
Actor	Usuario
Precondiciones	La herramienta muestra al usuario la página inicial de la sección de definición de reglas de juego.
Flujo básico	1. El usuario escoge un elemento de reglas del submenú que hay presente en la página. 2. El sistema muestra el formulario de creación del elemento de reglas. 3. El usuario pulsa el botón de cargar elemento de reglas. 4. El sistema ofrece una ventana modal al usuario con la lista de elementos de reglas guardados y un pequeño campo de búsqueda. 5. El usuario borra el elemento que desee pulsando el botón de borrar correspondiente. 6. El sistema elimina de la base de datos el elemento elegido. 7. El sistema cierra la ventana modal y devuelve la navegación a la página con el formulario de creación del elemento de reglas.
Flujo alternativo	5. a) El usuario pulsa el botón de cierre de la ventana. 1. El sistema cierra la ventana modal con el formulario. 2. Se muestra de nuevo el formulario de creación del elemento de reglas.
Flujo alternativo	5. b) El usuario introduce el nombre de un elemento en el campo de búsqueda y pulsa el botón de buscar. 1. El sistema muestra una nueva lista con los resultados de la búsqueda. 2. El usuario borra el elemento que desee pulsando el botón de borrar correspondiente. 3. Se retoma el flujo básico en el paso 6.
Postcondiciones	La herramienta carga la página con el formulario de creación del elemento de reglas correspondiente.

Tabla 94. Caso de uso CU-06

Identificador	CU-07
Nombre	Editar elemento reglas
Descripción	El usuario edita un elemento de reglas, modificando los campos que crea convenientes.
Actor	Usuario

Precondiciones	**La herramienta muestra al usuario la página inicial de la sección de definición de reglas de juego.**
Flujo básico	**1. El usuario escoge un elemento de reglas del submenú que hay presente en la página.** **2. El sistema muestra el formulario de creación del elemento de reglas.** **3. El usuario pulsa el botón de cargar elemento de reglas.** **4. El sistema ofrece una ventana modal al usuario con la lista de elementos de reglas guardados y un pequeño campo de búsqueda.** **5. El usuario selecciona el elemento de reglas que desee y pulsa el botón de cargar.** **6. El sistema cierra la ventana modal y muestra de nuevo el formulario de creación del elemento, pero esta vez con los campos rellenados con los datos recuperados desde base de datos.** **7. El usuario modifica los campos que desee, añade o quita los subelementos que considere necesarios y pulsa el botón de guardar.** **8. El sistema almacena en la base de datos el elemento modificado por el usuario.** **9. El sistema muestra un mensaje notificando el correcto guardado del elemento de reglas.** **10. El sistema recarga la página.**
Flujo alternativo	**5. a) El usuario pulsa el botón de cierre de la ventana.** **1. El sistema cierra la ventana modal con el formulario.** **2. Se muestra de nuevo el formulario de creación del elemento de reglas.**
Flujo alternativo	**5. b) El usuario introduce el nombre de un elemento en el campo de búsqueda y pulsa el botón de buscar.** **1. El sistema muestra una nueva lista con los resultados de la búsqueda.** **2. El usuario selecciona el elemento de reglas que desee y pulsa el botón de cargar.** **3. Se retoma el flujo básico en el paso 6.**
Flujo alternativo	**8. a) El usuario no rellena todos los campos obligatorios y pulsa el botón de guardar.** **1. El sistema le indica al usuario los campos que son obligatorios y no ejecuta ninguna acción de guardado.** **2. Se retoma el flujo básico en el inicio del paso 7.**
Flujo alternativo	**8. b) El usuario rellena de manera incorrecta los campos del formulario y pulsa el botón de guardar.** **1. El sistema le indica al usuario los campos erróneos y proporciona una ayuda para su corrección.** **2. Se retoma el flujo básico en el inicio del paso 7.**
Postcondiciones	**La herramienta muestra la página con el formulario de creación del elemento de reglas vacío.**

Tabla 95. Caso de uso CU-07

Identificador	CU-08
Nombre	Crear conjunto reglas
Descripción	El usuario crea un conjunto de reglas rellenando los campos del formulario asociado y añadiendo los elementos que considere oportunos.
Actor	Usuario
Precondiciones	• El usuario escoge en el submenú la opción de crear reglas. • La herramienta muestra al usuario la página con el formulario de creación de reglas.
Flujo básico	1. El usuario rellena todos los campos necesarios y ejecuta las acciones de agregar elementos. 2. El usuario pulsa el botón de guardar. 3. El sistema almacena en la base de datos el conjunto de reglas creado por el usuario. 4. El sistema muestra un mensaje notificando el correcto guardado del conjunto de reglas. 5. El sistema recarga la página.
Flujo alternativo	2. a) El usuario no rellena todos los campos obligatorios y pulsa el botón de guardar. 1. El sistema le indica al usuario los campos que son obligatorios y no ejecuta ninguna acción de guardado. 2. Se retoma el flujo básico en el inicio del paso 1.
Flujo alternativo	2. b) El usuario rellena de manera incorrecta los campos del formulario y pulsa el botón de guardar. 1. El sistema le indica al usuario los campos erróneos y proporciona una ayuda para su corrección. 2. Se retoma el flujo básico en el inicio del paso 1.
Postcondiciones	La herramienta muestra la página con el formulario de creación de reglas vacío.

Tabla 96. Caso de uso CU-08

Identificador	CU-09
Nombre	Borrar conjunto reglas
Descripción	El usuario borra un conjunto de reglas, desapareciendo éste de la base de datos junto con todos sus elementos asociados.
Actor	Usuario
Precondiciones	• El usuario escoge en el submenú la opción de crear reglas. • La herramienta muestra al usuario la página con el formulario de creación de reglas.
Flujo básico	1. El usuario pulsa el botón de cargar reglas. 2. El sistema ofrece una ventana modal al usuario con la

	lista de reglas guardadas y un pequeño campo de búsqueda. **3. El usuario borra el conjunto de reglas que desee pulsando el botón de borrar correspondiente.** **4. El sistema elimina de la base de datos el conjunto de reglas elegido.** **5. El sistema cierra la ventana modal y devuelve la navegación a la página con el formulario de creación del conjunto de reglas.**
Flujo alternativo	**3. a) El usuario pulsa el botón de cierre de la ventana.** **1. El sistema cierra la ventana modal con el formulario.** **2. Se muestra de nuevo el formulario de creación del conjunto de reglas.**
Flujo alternativo	**3. b) El usuario introduce el nombre de un conjunto de reglas en el campo de búsqueda y pulsa el botón de buscar.** **1. El sistema muestra una nueva lista con los resultados de la búsqueda.** **2. El usuario borra el conjunto de reglas que desee pulsando el botón de borrar correspondiente.** **3. Se retoma el flujo básico en el paso 3.**
Postcondiciones	**La herramienta carga la página con el formulario de creación del conjunto de reglas correspondiente.**

Tabla 97. Caso de uso CU-09

Identificador	**CU-10**
Nombre	**Editar conjunto reglas**
Descripción	**El usuario edita un conjunto de reglas, modificando los campos que crea convenientes.**
Actor	**Usuario**
Precondiciones	**• El usuario escoge en el submenú la opción de crear reglas.** **• La herramienta muestra al usuario la página con el formulario de creación de reglas.**
Flujo básico	**1. El usuario pulsa el botón de cargar reglas.** **2. El sistema ofrece una ventana modal al usuario con la lista de reglas guardadas y un pequeño campo de búsqueda.** **3. El usuario selecciona el conjunto de reglas que desee y pulsa el botón de cargar.** **4. El sistema cierra la ventana modal y muestra de nuevo el formulario de creación del conjunto de reglas, pero esta vez con los campos rellenados con los datos recuperados desde base de datos.** **5. El usuario modifica los campos que desee, añade o quita**

	los elementos que considere necesarios y pulsa el botón de guardar. 6. El sistema almacena en la base de datos el conjunto de reglas modificado por el usuario. 7. El sistema muestra un mensaje notificando el correcto guardado del conjunto de reglas. 8. El sistema recarga la página.
Flujo alternativo	3. a) El usuario pulsa el botón de cierre de la ventana. 1. El sistema cierra la ventana modal con el formulario. 2. Se muestra de nuevo el formulario de creación del elemento de reglas.
Flujo alternativo	3. b) El usuario introduce el nombre de un conjunto de reglas en el campo de búsqueda y pulsa el botón de buscar. 1. El sistema muestra una nueva lista con los resultados de la búsqueda. 2. El usuario selecciona el conjunto de reglas que desee y pulsa el botón de cargar. 3. Se retoma el flujo básico en el paso 4.
Flujo alternativo	5. a) El usuario no rellena todos los campos obligatorios y pulsa el botón de guardar. 1. El sistema le indica al usuario los campos que son obligatorios y no ejecuta ninguna acción de guardado. 2. Se retoma el flujo básico en el inicio del paso 5.
Flujo alternativo	5. b) El usuario rellena de manera incorrecta los campos del formulario y pulsa el botón de guardar. 1. El sistema le indica al usuario los campos erróneos y proporciona una ayuda para su corrección. 2. Se retoma el flujo básico en el inicio del paso 5.
Postcondiciones	La herramienta muestra la página con el formulario de creación de reglas vacío.

Tabla 98. Caso de uso CU-10

Identificador	CU-11
Nombre	Crear subelemento escenario
Descripción	El usuario crea un subelemento de escenario rellenando los campos necesarios del formulario asociado.
Actor	Usuario
Precondiciones	La herramienta muestra al usuario la página inicial de la sección de definición de escenarios.
Flujo básico	1. El usuario escoge un elemento de escenario del submenú que hay presente en la página. 2. El sistema muestra el formulario de creación del elemento de escenario.

	3. El usuario pulsa el botón de crear subelemento de escenario. 4. El sistema ofrece una ventana modal al usuario con el formulario de creación del subelemento. 5. El usuario rellena todos los campos necesarios y pulsa el botón de guardar. 6. El sistema almacena en la base de datos el subelemento creado por el usuario. 7. El sistema cierra la ventana modal y devuelve la navegación a la página con el formulario de creación del elemento de escenario.
Flujo alternativo	5. a) El usuario pulsa el botón de cierre de la ventana. 1. El sistema cierra la ventana modal con el formulario. 2. Se muestra de nuevo el formulario de creación del elemento de escenario.
Flujo alternativo	5. b) El usuario no rellena todos los campos obligatorios y pulsa el botón de guardar. 1. El sistema le indica al usuario los campos que son obligatorios y no ejecuta ninguna acción de guardado. 2. Se retoma el flujo básico en el inicio del paso 5.
Flujo alternativo	5. c) El usuario rellena de manera incorrecta los campos del formulario y pulsa el botón de guardar. 1. El sistema le indica al usuario los campos erróneos y proporciona una ayuda para su corrección. 2. Se retoma el flujo básico en el inicio del paso 5.
Postcondiciones	La herramienta carga la página con el formulario de creación del elemento de escenario correspondiente.

Tabla 99. Caso de uso CU-11

Identificador	CU-12
Nombre	Borrar subelemento escenario
Descripción	El usuario borra un subelemento de escenario, desapareciendo éste de la base de datos.
Actor	Usuario
Precondiciones	La herramienta muestra al usuario la página inicial de la sección de definición de escenarios.
Flujo básico	1. El usuario escoge un elemento de escenario del submenú que hay presente en la página. 2. El sistema muestra el formulario de creación del elemento de escenario. 3. El usuario pulsa el botón de agregar subelemento de escenario. 4. El sistema ofrece una ventana modal al usuario con la

	lista de subelementos de escenario guardados y un pequeño campo de búsqueda. 5. El usuario borra el subelemento que desee pulsando el botón de borrar correspondiente. 6. El sistema elimina de la base de datos el subelemento elegido. 7. El sistema cierra la ventana modal y devuelve la navegación a la página con el formulario de creación del elemento de escenario.
Flujo alternativo	5. a) El usuario pulsa el botón de cierre de la ventana. 1. El sistema cierra la ventana modal con el formulario. 2. Se muestra de nuevo el formulario de creación del elemento de escenario.
Flujo alternativo	5. b) El usuario introduce el nombre de un subelemento en el campo de búsqueda y pulsa el botón de buscar. 4. El sistema muestra una nueva lista con los resultados de la búsqueda. 5. El usuario borra el subelemento que desee pulsando el botón de borrar correspondiente. 6. Se retoma el flujo básico en el paso 6.
Postcondiciones	La herramienta carga la página con el formulario de creación del elemento de escenario correspondiente.

Tabla 100. Caso de uso CU-12

Identificador	**CU-13**
Nombre	**Editar subelemento escenario**
Descripción	**El usuario edita un subelemento de escenario, modificando los campos que crea convenientes.**
Actor	**Usuario**
Precondiciones	**La herramienta muestra al usuario la página inicial de la sección de definición de escenarios.**
Flujo básico	1. El usuario escoge un elemento de escenario del submenú que hay presente en la página. 2. El sistema muestra el formulario de creación del elemento de escenario. 3. El usuario pulsa el botón de agregar subelemento de escenario. 4. El sistema ofrece una ventana modal al usuario con la lista de subelementos de escenario guardados y un pequeño campo de búsqueda. 5. El usuario pulsa el botón de editar correspondiente al subelemento elegido. 6. El sistema ofrece una ventana modal al usuario con el formulario de modificación del subelemento.

	7. El usuario rellena o modifica los campos que él decida y pulsa el botón de guardar. 8. El sistema almacena en la base de datos el subelemento modificado por el usuario. 9. El sistema cierra la ventana modal y devuelve la navegación a la página con el formulario de creación del elemento de escenario.
Flujo alternativo	5. a) El usuario pulsa el botón de cierre de la ventana. 1. El sistema cierra la ventana modal con el formulario. 2. Se muestra de nuevo el formulario de creación del elemento de escenario.
Flujo alternativo	5. b) El usuario introduce el nombre de un subelemento en el campo de búsqueda y pulsa el botón de buscar. 1. El sistema muestra una nueva lista con los resultados de la búsqueda. 2. El usuario pulsa el botón de editar correspondiente al subelemento elegido. 3. Se retoma el flujo básico en el paso 6.
Flujo alternativo	5. a) El usuario pulsa el botón de cierre de la ventana. 1. El sistema cierra la ventana modal con el formulario. 2. Se muestra de nuevo el formulario de creación del elemento de escenario.
Flujo alternativo	7. b) El usuario no rellena todos los campos obligatorios y pulsa el botón de guardar. 1. El sistema le indica al usuario los campos que son obligatorios y no ejecuta ninguna acción de guardado. 2. Se retoma el flujo básico en el inicio del paso 7.
Flujo alternativo	7. c) El usuario rellena de manera incorrecta los campos del formulario y pulsa el botón de guardar. 1. El sistema le indica al usuario los campos erróneos y proporciona una ayuda para su corrección. 2. Se retoma el flujo básico en el inicio del paso 7.
Postcondiciones	La herramienta carga la página con el formulario de creación del elemento de escenario correspondiente.

Tabla 101. Caso de uso CU-13

Identificador	**CU-14**
Nombre	**Agregar subelemento escenario**
Descripción	**El usuario agrega un subelemento de escenario al correspondiente elemento de escenario al que pertenece.**
Actor	**Usuario**

Precondiciones	**La herramienta muestra al usuario la página inicial de la sección de definición de escenarios.**
Flujo básico	**1. El usuario escoge un elemento de escenario del submenú que hay presente en la página.** **2. El sistema muestra el formulario de creación del elemento de escenario.** **3. El usuario pulsa el botón de agregar subelemento de escenario.** **4. El sistema ofrece una ventana modal al usuario con la lista de subelementos de escenario guardados y un pequeño campo de búsqueda.** **5. El usuario selecciona el subelemento de escenario que desee y pulsa el botón de agregar.** **6. El sistema cierra la ventana modal y devuelve la navegación a la página con el formulario de creación del elemento de escenario.**
Flujo alternativo	**5. a) El usuario pulsa el botón de cierre de la ventana.** **1. El sistema cierra la ventana modal con el formulario** **2. Se muestra de nuevo el formulario de creación del elemento de escenario.**
Flujo alternativo	**5. b) El usuario introduce el nombre de un subelemento en el campo de búsqueda y pulsa el botón de buscar.** **1. El sistema muestra una nueva lista con los resultados de la búsqueda.** **2. El usuario selecciona el subelemento de escenario que desee y pulsa el botón de agregar.** **3. Se retoma el flujo básico en el paso 6.**
Postcondiciones	**La herramienta carga la página con el subelemento incluido en el formulario de creación del elemento de escenario correspondiente.**

Tabla 102. Caso de uso CU-14

Identificador	**CU-15**
Nombre	**Crear elemento escenario**
Descripción	**El usuario crea un elemento de escenario rellenando los campos necesarios del formulario asociado.**
Actor	**Usuario**
Precondiciones	**La herramienta muestra al usuario la página inicial de la sección de definición de escenarios.**
Flujo básico	**1. El usuario escoge un elemento de escenario del submenú que hay presente en la página.** **2. El sistema muestra el formulario de creación del elemento de escenario.** **3. El usuario rellena todos los campos necesarios y ejecuta**

	las acciones de agregar subelementos en caso de que los haya. 4. El usuario pulsa el botón de guardar. 5. El sistema almacena en la base de datos el elemento creado por el usuario. 6. El sistema muestra un mensaje notificando el correcto guardado del elemento de escenario. 7. El sistema recarga la página.
Flujo alternativo	3. a) El usuario no rellena todos los campos obligatorios y pulsa el botón de guardar. 1. El sistema le indica al usuario los campos que son obligatorios y no ejecuta ninguna acción de guardado. 2. Se retoma el flujo básico en el inicio del paso 3.
Flujo alternativo	3. b) El usuario rellena de manera incorrecta los campos del formulario y pulsa el botón de guardar. 1. El sistema le indica al usuario los campos erróneos y proporciona una ayuda para su corrección. 2. Se retoma el flujo básico en el inicio del paso 3.
Postcondiciones	La herramienta muestra la página con el formulario de creación del elemento de escenario vacío.

Tabla 103. Caso de uso CU-15

Identificador	CU-16
Nombre	Borrar elemento escenario
Descripción	El usuario borra un elemento de escenario, desapareciendo éste de la base de datos junto con todos sus subelementos asociados.
Actor	Usuario
Precondiciones	La herramienta muestra al usuario la página inicial de la sección de definición de escenarios.
Flujo básico	1. El usuario escoge un elemento de escenario del submenú que hay presente en la página. 2. El sistema muestra el formulario de creación del elemento de escenario. 3. El usuario pulsa el botón de cargar elemento de escenario. 4. El sistema ofrece una ventana modal al usuario con la lista de elementos de escenario guardados y un pequeño campo de búsqueda. 5. El usuario borra el elemento que desee pulsando el botón de borrar correspondiente. 6. El sistema elimina de la base de datos el elemento elegido. 7. El sistema cierra la ventana modal y devuelve la navegación a la página con el formulario de creación del elemento de escenario.

Flujo alternativo	**5. a) El usuario pulsa el botón de cierre de la ventana.** **1. El sistema cierra la ventana modal con el formulario.** **2. Se muestra de nuevo el formulario de creación del elemento de escenario.**
Flujo alternativo	**5. b) El usuario introduce el nombre de un elemento en el campo de búsqueda y pulsa el botón de buscar.** **1. El sistema muestra una nueva lista con los resultados de la búsqueda.** **2. El usuario borra el elemento que desee pulsando el botón de borrar correspondiente.** **3. Se retoma el flujo básico en el paso 6.**
Postcondiciones	**La herramienta carga la página con el formulario de creación del elemento de escenario correspondiente.**

Tabla 104. Caso de uso CU-16

Identificador	**CU-17**
Nombre	**Editar elemento escenario**
Descripción	**El usuario edita un elemento de escenario, modificando los campos que crea convenientes.**
Actor	**Usuario**
Precondiciones	**La herramienta muestra al usuario la página inicial de la sección de definición de escenarios.**
Flujo básico	**1. El usuario escoge un elemento de escenario del submenú que hay presente en la página.** **2. El sistema muestra el formulario de creación del elemento de escenario.** **3. El usuario pulsa el botón de cargar elemento de escenario.** **4. El sistema ofrece una ventana modal al usuario con la lista de elementos de escenario guardados y un pequeño campo de búsqueda.** **5. El usuario selecciona el elemento de escenario que desee y pulsa el botón de cargar.** **6. El sistema cierra la ventana modal y muestra de nuevo el formulario de creación del elemento, pero esta vez con los campos rellenados con los datos recuperados desde base de datos.** **7. El usuario modifica los campos que desee, añade o quita los subelementos que considere necesarios y pulsa el botón de guardar.** **8. El sistema almacena en la base de datos el elemento modificado por el usuario.** **9. El sistema muestra un mensaje notificando el correcto guardado del elemento de escenario.**

	10. El sistema recarga la página.
Flujo alternativo	**5. a) El usuario pulsa el botón de cierre de la ventana.** **1. El sistema cierra la ventana modal con el formulario.** **2. Se muestra de nuevo el formulario de creación del elemento de escenario.**
Flujo alternativo	**5. b) El usuario introduce el nombre de un elemento en el campo de búsqueda y pulsa el botón de buscar.** **1. El sistema muestra una nueva lista con los resultados de la búsqueda.** **2. El usuario selecciona el elemento de escenario que desee y pulsa el botón de cargar.** **3. Se retoma el flujo básico en el paso 6.**
Flujo alternativo	**7. a) El usuario no rellena todos los campos obligatorios y pulsa el botón de guardar.** **1. El sistema le indica al usuario los campos que son obligatorios y no ejecuta ninguna acción de guardado.** **2. Se retoma el flujo básico en el inicio del paso 7.**
Flujo alternativo	**7. b) El usuario rellena de manera incorrecta los campos del formulario y pulsa el botón de guardar.** **1. El sistema le indica al usuario los campos erróneos y proporciona una ayuda para su corrección.** **2. Se retoma el flujo básico en el inicio del paso 7.**
Postcondiciones	**La herramienta muestra la página con el formulario de creación del elemento de escenario vacío.**

Tabla 105. Caso de uso CU-17

Identificador	**CU-18**
Nombre	**Crear escenario**
Descripción	**El usuario crea un escenario rellenando los campos del formulario asociado y añadiendo los elementos que considere oportunos.**
Actor	**Usuario**
Precondiciones	• **El usuario escoge en el submenú la opción de crear escenarios.** • **La herramienta muestra al usuario la página con el formulario de creación de escenarios.**
Flujo básico	**1. El usuario rellena todos los campos necesarios y ejecuta las acciones de agregar elementos.** **2. El usuario pulsa el botón de guardar.** **3. El sistema almacena en la base de datos el escenario creado por el usuario.** **4. El sistema muestra un mensaje notificando el correcto**

	guardado del escenario. 5. El sistema recarga la página.
Flujo alternativo	1. a) El usuario no rellena todos los campos obligatorios y pulsa el botón de guardar. 1. El sistema le indica al usuario los campos que son obligatorios y no ejecuta ninguna acción de guardado. 2. Se retoma el flujo básico en el inicio del paso 1.
Flujo alternativo	1. b) El usuario rellena de manera incorrecta los campos del formulario y pulsa el botón de guardar. 1. El sistema le indica al usuario los campos erróneos y proporciona una ayuda para su corrección. 2. Se retoma el flujo básico en el inicio del paso 1.
Postcondiciones	La herramienta muestra la página con el formulario de creación de escenarios vacío.

Tabla 106. Caso de uso CU-18

Identificador	CU-19
Nombre	Borrar escenario
Descripción	El usuario borra un escenario, desapareciendo éste de la base de datos junto con todos sus elementos asociados.
Actor	Usuario
Precondiciones	• El usuario escoge en el submenú la opción de crear escenarios. • La herramienta muestra al usuario la página con el formulario de creación de escenarios.
Flujo básico	1. El usuario pulsa el botón de cargar escenarios. 2. El sistema ofrece una ventana modal al usuario con la lista de escenarios guardados y un pequeño campo de búsqueda. 3. El usuario borra el escenario que desee pulsando el botón de borrar correspondiente. 4. El sistema elimina de la base de datos el escenario elegido. 5. El sistema cierra la ventana modal y devuelve la navegación a la página con el formulario de creación del escenario.
Flujo alternativo	3. a) El usuario pulsa el botón de cierre de la ventana. 1. El sistema cierra la ventana modal con el formulario. 2. Se muestra de nuevo el formulario de creación del escenario.
Flujo alternativo	3. b) El usuario introduce el nombre de un escenario en el campo de búsqueda y pulsa el botón de buscar. 1. El sistema muestra una nueva lista con los

	resultados de la búsqueda. **2. El usuario borra el escenario que desee pulsando el botón de borrar correspondiente.** **3. Se retoma el flujo básico en el paso 3.**
Postcondiciones	**La herramienta carga la página con el formulario de creación del escenario correspondiente.**

Tabla 107. Caso de uso CU-19

Identificador	**CU-20**
Nombre	**Editar escenario**
Descripción	**El usuario edita un escenario, modificando los campos que crea convenientes.**
Actor	**Usuario**
Precondiciones	• **El usuario escoge en el submenú la opción de crear escenarios.** • **La herramienta muestra al usuario la página con el formulario de creación de escenarios.**
Flujo básico	**1. El usuario pulsa el botón de cargar escenarios.** **2. El sistema ofrece una ventana modal al usuario con la lista de escenarios guardados y un pequeño campo de búsqueda.** **3. El usuario selecciona el escenario que desee y pulsa el botón de cargar.** **4. El sistema cierra la ventana modal y muestra de nuevo el formulario de creación del escenario, pero esta vez con los campos rellenados con los datos recuperados desde base de datos.** **5. El usuario modifica los campos que desee, añade o quita los elementos que considere necesarios y pulsa el botón de guardar.** **6. El sistema almacena en la base de datos el escenario modificado por el usuario.** **7. El sistema muestra un mensaje notificando el correcto guardado del escenario.** **8. El sistema recarga la página.**
Flujo alternativo	**3. a) El usuario pulsa el botón de cierre de la ventana.** **1. El sistema cierra la ventana modal con el formulario.** **2. Se muestra de nuevo el formulario de creación del escenario.**
Flujo alternativo	**3. b) El usuario introduce el nombre de un escenario en el campo de búsqueda y pulsa el botón de buscar.** **1. El sistema muestra una nueva lista con los resultados de la búsqueda.** **2. El usuario selecciona el escenario que desee y pulsa**

	el botón de cargar. 3. Se retoma el flujo básico en el paso 4.
Flujo alternativo	5. a) El usuario no rellena todos los campos obligatorios y pulsa el botón de guardar. 1. El sistema le indica al usuario los campos que son obligatorios y no ejecuta ninguna acción de guardado. 2. Se retoma el flujo básico en el inicio del paso 5.
Flujo alternativo	5. b) El usuario rellena de manera incorrecta los campos del formulario y pulsa el botón de guardar. 1. El sistema le indica al usuario los campos erróneos y proporciona una ayuda para su corrección. 2. Se retoma el flujo básico en el inicio del paso 5.
Postcondiciones	La herramienta muestra la página con el formulario de creación de escenarios vacío.

Tabla 108. Caso de uso CU-20

Identificador	CU-21
Nombre	Crear match
Descripción	El usuario crea un match rellenando los campos del formulario asociado y añadiendo los un conjunto de reglas y un escenario.
Actor	Usuario
Precondiciones	La herramienta muestra al usuario la página inicial de la sección de definición de videojuegos.
Flujo básico	1. El usuario rellena todos los campos necesarios. 2. El usuario pulsa el botón de cargar reglas. 3. El sistema ofrece una ventana modal al usuario con la lista de reglas guardadas y un pequeño campo de búsqueda. 4. El usuario selecciona el conjunto de reglas que desee y pulsa el botón de cargar. 5. El sistema cierra la ventana modal y muestra de nuevo el formulario de creación del match, pero esta vez con el conjunto de reglas incluido. 6. El usuario pulsa el botón de cargar escenarios. 7. El sistema ofrece una ventana modal al usuario con la lista de escenarios guardados y un pequeño campo de búsqueda. 8. El usuario selecciona el escenario que desee y pulsa el botón de cargar. 9. El sistema cierra la ventana modal y muestra de nuevo el formulario de creación del match, pero esta vez con el escenario incluido. 10. El usuario pulsa el botón de guardar. 11. El sistema almacena en la base de datos el match creado

	por el usuario. 12. El sistema muestra un mensaje notificando el correcto guardado del match. 13. El sistema recarga la página.
Flujo alternativo	1. a) El usuario no rellena todos los campos obligatorios y pulsa el botón de guardar. 1. El sistema le indica al usuario los campos que son obligatorios y no ejecuta ninguna acción de guardado. 2. Se retoma el flujo básico en el inicio del paso 1.
Flujo alternativo	1. b) El usuario rellena de manera incorrecta los campos del formulario y pulsa el botón de guardar. 1. El sistema le indica al usuario los campos erróneos y proporciona una ayuda para su corrección. 2. Se retoma el flujo básico en el inicio del paso 1.
Flujo alternativo	4. a) El usuario pulsa el botón de cierre de la ventana. 1. El sistema cierra la ventana modal con el formulario. 2. Se muestra de nuevo el formulario de creación del escenario.
Flujo alternativo	4. b) El usuario introduce el nombre de un conjunto de reglas en el campo de búsqueda y pulsa el botón de buscar. 1. El sistema muestra una nueva lista con los resultados de la búsqueda. 2. El usuario selecciona el conjunto de reglas que desee y pulsa el botón de cargar. 3. Se retoma el flujo básico en el paso 5.
Flujo alternativo	8. a) El usuario pulsa el botón de cierre de la ventana. 1. El sistema cierra la ventana modal con el formulario. 2. Se muestra de nuevo el formulario de creación del escenario.
Flujo alternativo	8. b) El usuario introduce el nombre de un escenario en el campo de búsqueda y pulsa el botón de buscar. 1. El sistema muestra una nueva lista con los resultados de la búsqueda. 2. El usuario selecciona el escenario que desee y pulsa el botón de cargar. 3. Se retoma el flujo básico en el paso 9.
Postcondiciones	La herramienta muestra la página con el formulario de creación de matches vacío.

Tabla 109. Caso de uso CU-21

Identificador	CU-22
Nombre	Borrar match
Descripción	El usuario borra un match, desapareciendo éste de la base de datos.
Actor	Usuario
Precondiciones	La herramienta muestra al usuario la página inicial de la sección de definición de videojuegos.
Flujo básico	1. El usuario pulsa el botón de cargar matches. 2. El sistema ofrece una ventana modal al usuario con la lista de matches guardados y un pequeño campo de búsqueda. 3. El usuario borra el match que desee pulsando el botón de borrar correspondiente. 4. El sistema elimina de la base de datos el match elegido. 5. El sistema cierra la ventana modal y devuelve la navegación a la página con el formulario de creación del match.
Flujo alternativo	3. a) El usuario pulsa el botón de cierre de la ventana. 1. El sistema cierra la ventana modal con el formulario. 2. Se muestra de nuevo el formulario de creación del match.
Flujo alternativo	3. b) El usuario introduce el nombre de un match en el campo de búsqueda y pulsa el botón de buscar. 1. El sistema muestra una nueva lista con los resultados de la búsqueda. 2. El usuario borra el match que desee pulsando el botón de borrar correspondiente. 3. Se retoma el flujo básico en el paso 3.
Postcondiciones	La herramienta carga la página con el formulario de creación del match correspondiente.

Tabla 110. Caso de uso CU-22

Identificador	CU-23
Nombre	Generar XML a partir de match
Descripción	El usuario genera un documento XML con el diseño del videojuego a partir de un match cargado desde base de datos.
Actor	Usuario
Precondiciones	La herramienta muestra al usuario la página inicial de la sección de definición de videojuegos.
Flujo básico	1. El usuario pulsa el botón de cargar matches. 2. El sistema ofrece una ventana modal al usuario con la

	lista de matches guardados y un pequeño campo de búsqueda. **3. El usuario selecciona el match que desee y pulsa el botón de cargar.** **4. El sistema cierra la ventana modal y muestra de nuevo el formulario de creación del match, pero esta vez con los campos rellenados con los datos recuperados desde base de datos más un campo extra con la ruta de destino del XML.** **5. El usuario rellena el campo extra y pulsa el botón de generar XML.** **6. El sistema muestra un mensaje notificando la correcta generación del XML.** **7. El sistema recarga la página.** **8. El usuario comprueba la correcta creación del XML en la ruta especificada.**
Flujo alternativo	**3. a) El usuario pulsa el botón de cierre de la ventana.** **1. El sistema cierra la ventana modal con el formulario.** **2. Se muestra de nuevo el formulario de creación del match.**
Flujo alternativo	**3. b) El usuario introduce el nombre de un match en el campo de búsqueda y pulsa el botón de buscar.** **1. El sistema muestra una nueva lista con los resultados de la búsqueda.** **2. El usuario selecciona el match que desee y pulsa el botón de cargar.** **3. Se retoma el flujo básico en el paso 4.**
Flujo alternativo	**5. a) El usuario no rellena todos los campos obligatorios y pulsa el botón de generar XML.** **1. El sistema le indica al usuario los campos que son obligatorios y no ejecuta ninguna acción de generación del XML.** **2. Se retoma el flujo básico en el inicio del paso 5.**
Flujo alternativo	**5. b) El usuario rellena de manera incorrecta los campos del formulario y pulsa el botón de guardar.** **1. El sistema le indica al usuario los campos erróneos y proporciona una ayuda para su corrección.** **2. Se retoma el flujo básico en el inicio del paso 5.**
Postcondiciones	**El documento XML se crea en la ruta que ha especificado el usuario con la información correcta.**

Tabla 111. Caso de uso CU-23

Identificador	**CU-24**
Nombre	**Generar XML a partir de storytellings**
Descripción	**La herramienta muestra al usuario la página inicial de la sección de definición de videojuegos.**
Actor	**Usuario**
Precondiciones	• **El usuario escoge en el submenú la opción de crear reglas.** • **La herramienta muestra al usuario la página con el formulario de creación de reglas.**
Flujo básico	1. **El usuario pulsa sobre el botón de crear mini-juegos.** 2. **El sistema muestra una nueva página en la que se pueden añadir storytellings.** 3. **El sistema ofrece una ventana modal al usuario con la lista de storytellings guardados y un pequeño campo de búsqueda.** 4. **El usuario selecciona los storytellings que desee y pulsa el botón de cargar.** 5. **El sistema cierra la ventana modal y muestra de nuevo la página anteriormente mencionada, pero esta vez con los storytellings cargados más un campo extra con la ruta de destino del XML.** 6. **El usuario rellena el campo extra y pulsa el botón de generar XML.** 7. **El sistema muestra un mensaje notificando la correcta generación del XML.** 8. **El sistema recarga la página.** 9. **El usuario comprueba la correcta creación del XML en la ruta especificada.**
Flujo alternativo	4. **a) El usuario pulsa el botón de cierre de la ventana.** 1. **El sistema cierra la ventana modal con el formulario.** 2. **Se muestra de nuevo la página de generación del XML.**
Flujo alternativo	4. **b) El usuario introduce el nombre de un storytelling en el campo de búsqueda y pulsa el botón de buscar.** 1. **El sistema muestra una nueva lista con los resultados de la búsqueda.** 2. **El usuario selecciona el storytelling que desee y pulsa el botón de cargar.** 3. **Se retoma el flujo básico en el paso 4.**
Flujo alternativo	6. **a) El usuario no rellena todos los campos obligatorios y pulsa el botón de generar XML.** 1. **El sistema le indica al usuario los campos que son obligatorios y no ejecuta ninguna acción de generar XML.** 2. **Se retoma el flujo básico en el inicio del paso 6.**
Flujo alternativo	6. **b) El usuario rellena de manera incorrecta los campos del formulario y pulsa el botón de generar XML.**

	1. El sistema le indica al usuario los campos erróneos y proporciona una ayuda para su corrección. **2. Se retoma el flujo básico en el inicio del paso 6.**
Postcondiciones	**La herramienta muestra la página de generación del XML vacía de storytellings.**

Tabla 112. Caso de uso CU-24

4.5 Matrices de trazabilidad

En un proyecto es complicado comprobar si todos los requisitos de usuario quedan cubiertos por los casos de usos y los requisitos de software especificados. Las matrices de trazabilidad son herramientas que se utilizan para saber qué requisitos quedan perfectamente cubiertos y cuáles no.

Las matrices de trazabilidad son sencillas de utilizar y facilitan de un simple vistazo qué requisitos no quedan respaldados y, por tanto, es necesario, buscar un modo para que lo sean. También ayudan a detectar dependencias entre requisitos, por lo que no es complicado averiguar cuáles se verían afectados al modificar o eliminar alguno.

A continuación, se presentan las matrices de trazabilidad entre requisitos de usuario y casos de uso, requisitos de software funcionales y no funcionales:

REQUISITOS DE USUARIO	CASOS DE USO																						
REQUISITOS DE CAPACIDAD	CU-01	CU-02	CU-03	CU-04	CU-05	CU-06	CU-07	CU-08	CU-09	CU-10	CU-11	CU-12	CU-13	CU-14	CU-15	CU-16	CU-17	CU-18	CU-19	CU-20	CU-21	CU-22	CU-23	CU-24
RUC-01	X	X	X	X	X	X	X	X	X	X														
RUC-02											X	X	X	X	X	X	X	X	X	X				
RUC-03			X				X			X			X				X			X			X	
RUC-04	X				X			X			X				X			X			X			
RUC-05																					X		X	
RUC-06																								X
RUC-07																							X	
REQUISITOS DE RESTRICCIÓN	CU-01	CU-02	CU-03	CU-04	CU-05	CU-06	CU-07	CU-08	CU-09	CU-10	CU-11	CU-12	CU-13	CU-14	CU-15	CU-16	CU-17	CU-18	CU-19	CU-20	CU-21	CU-22	CU-23	CU-24
RUR-01	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
RUR-02	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
RUR-03	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
RUR-04	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
RUR-05	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
RUR-06	X		X		X		X	X		X	X		X		X		X	X		X	X		X	X
RUR-07				X				X						X				X			X			
RUR-08	X		X		X		X	X		X	X		X		X		X	X		X	X			

Tabla 113. Matriz de trazabilidad entre requisitos de usuario y casos de uso

REQUISITOS DE USUARIO	REQUISITOS DE SOFTWARE FUNCIONALES																	
REQUISITOS DE CAPACIDAD	**RSF-01**	**RSF-02**	**RSF-03**	**RSF-04**	**RSF-05**	**RSF-06**	**RSF-07**	**RSF-08**	**RSF-09**	**RSF-10**	**RSF-11**	**RSF-12**	**RSF-13**	**RSF-14**	**RSF-15**	**RSF-16**	**RSF-17**	**RSF-18**
RUC-01	X	X	X	X	X	X	X	X		X		X	X	X				
RUC-02																	X	X
RUC-03									X		X				X			
RUC-04		X	X	X	X	X				X		X						X
RUC-05																		
RUC-06																X		
RUC-07																X		
REQUISITOS DE CAPACIDAD	**RSF-19**	**RSF-20**	**RSF-21**	**RSF-22**	**RSF-23**	**RSF-24**	**RSF-25**	**RSF-26**	**RSF-27**	**RSF-28**	**RSF-29**	**RSF-30**	**RSF-31**	**RSF-32**	**RSF-33**	**RSF-34**	**RSF-35**	
RUC-01																		
RUC-02	X	X	X	X	X		X		X	X	X							
RUC-03						X		X				X			X			
RUC-04	X	X	X				X		X				X					
RUC-05																X		
RUC-06																		
RUC-07																	X	

Tabla 114. Matriz de trazabilidad entre requisitos de usuario (de capacidad) y de software funcionales

REQUISITOS DE USUARIO	REQUISITOS DE SOFTWARE FUNCIONALES																	
REQUISITOS DE RESTRICCIÓN	**RSF-01**	**RSF-02**	**RSF-03**	**RSF-04**	**RSF-05**	**RSF-06**	**RSF-07**	**RSF-08**	**RSF-09**	**RSF-10**	**RSF-11**	**RSF-12**	**RSF-13**	**RSF-14**	**RSF-15**	**RSF-16**	**RSF-17**	**RSF-18**
RUR-01		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		X
RUR-02		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		X
RUR-03		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		X
RUR-04		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		X
RUR-05		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		X
RUR-06		X	X	X	X	X	X			X		X	X			X		X
RUR-07		X	X	X	X	X	X					X	X					X
RUR-08		X	X	X	X	X	X			X		X	X					X
REQUISITOS DE RESTRICCIÓN	**RSF-19**	**RSF-20**	**RSF-21**	**RSF-22**	**RSF-23**	**RSF-24**	**RSF-25**	**RSF-26**	**RSF-27**	**RSF-28**	**RSF-29**	**RSF-30**	**RSF-31**	**RSF-32**	**RSF-33**	**RSF-34**	**RSF-35**	
RUR-01	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
RUR-02	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
RUR-03	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	

RUR-04	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
RUR-05	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
RUR-06	X	X	X	X			X		X	X			X				X	
RUR-07				X					X	X			X					
RUR-08	X	X	X	X			X		X	X			X					

Tabla 115. Matriz de trazabilidad entre requisitos de usuario (de restricción) y de software funcionales

REQUISITOS DE USUARIO	REQUISITOS DE SOFTWARE NO FUNCIONALES																							
REQUISITOS DE CAPACIDAD	**RSNF-01**	**RSNF-02**	**RSNF-03**	**RSNF-04**	**RSNF-05**	**RSNF-06**	**RSNF-07**	**RSNF-08**	**RSNF-09**	**RSNF-10**	**RSNF-11**	**RSNF-12**	**RSNF-13**	**RSNF-14**	**RSNF-15**	**RSNF-16**	**RSNF-17**	**RSNF-18**	**RSNF-19**	**RSNF-20**	**RSNF-21**	**RSNF-22**	**RSNF-23**	**RSNF-24**
RUC-01		X	X			X	X	X	X	X	X	X	X	X		X	X	X	X		X	X	X	X
RUC-02		X	X			X	X	X	X	X	X	X	X	X		X	X	X	X		X	X	X	X
RUC-03								X	X				X	X		X	X	X	X					
RUC-04											X													
RUC-05								X	X	X	X	X	X	X		X	X				X	X	X	X
RUC-06								X	X	X	X	X	X	X		X	X				X	X	X	X
RUC-07														X		X	X							
REQUISITOS DE RESTRICCIÓN	**RSNF-01**	**RSNF-02**	**RSNF-03**	**RSNF-04**	**RSNF-05**	**RSNF-06**	**RSNF-07**	**RSNF-08**	**RSNF-09**	**RSNF-10**	**RSNF-11**	**RSNF-12**	**RSNF-13**	**RSNF-14**	**RSNF-15**	**RSNF-16**	**RSNF-17**	**RSNF-18**	**RSNF-19**	**RSNF-20**	**RSNF-21**	**RSNF-22**	**RSNF-23**	**RSNF-24**
RUR-01															X									
RUR-02	X				X																			
RUR-03	X																							
RUR-04									X															
RUR-05								X				X								X	X	X		
RUR-06										X	X													
RUR-07										X														
RUR-08				X																				

Tabla 116. Matriz de trazabilidad entre requisitos de usuario y de software no funcionales

5. Diseño

En esta sección se va a llevar a cabo el diseño del sistema. Una vez que se han obtenido todos los requisitos que se han de cumplir, el siguiente paso consiste en definir el diseño de la herramienta de autoría.

En primer lugar, se ofrecerá una visión general del sistema. A continuación, se describirá la arquitectura física sobre la que se apoya el sistema. Tras esto, se indicarán varias cuestiones de diseño, como el patrón de diseño escogido o el modelo de base de datos. Por último, se detallará la arquitectura software y sus diferentes subsistemas y componentes.

5.1 Visión general del sistema

En la sección anterior se dio una descripción general del sistema, en la que se puso de manifiesto su funcionalidad. En esta subsección se pretende enfocar la funcionalidad desde el punto de vista de las acciones que tienen lugar para lograr el objetivo final, que es el diseño de un videojuego o parte de él. De este modo se tendrá una visión general del sistema para después comprender mejor la arquitectura y el diseño escogidos para el mismo.

El usuario principal de la herramienta va a ser un educador con un perfil técnico normalmente bajo. A través de las diferentes secciones que conformarán la herramienta, dicho usuario podrá crear los distintos tipos de información que necesitará para producir el diseño de su propio videojuego. En el siguiente esquema se muestran las diferentes acciones que se ejecutan en el proceso, desde que el usuario interactúa con la herramienta hasta que el motor gráfico Unity 3D crea el videojuego:

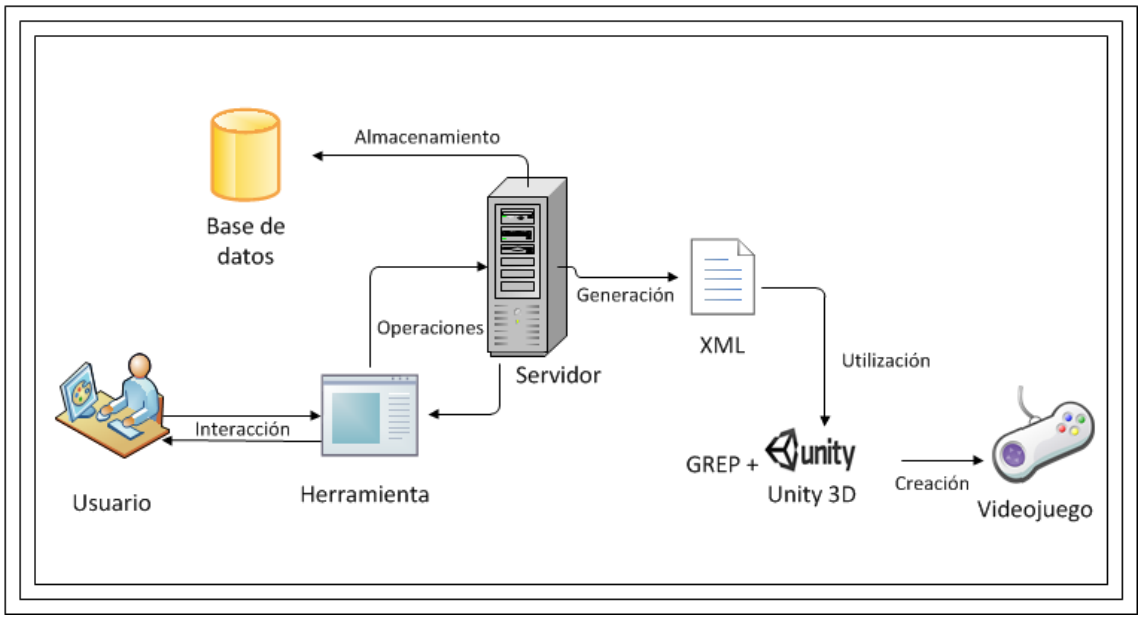


Ilustración 18. Visión general del sistema

Como se puede ver en el esquema, el usuario interactúa con la herramienta y ésta efectúa diversas operaciones con un servidor según lo solicitado por el usuario. El servidor se encarga de tramitar todas las operaciones recibidas por la herramienta y de

almacenar en una base de datos la información introducida por el usuario en la herramienta. Por otro lado, también se encarga de la generación del XML a partir de la información necesaria guardada en la base de datos. En último lugar, el documento XML con la información de diseño es utilizada por el motor GREP junto con Unity 3D para dar lugar al videojuego diseñado por el usuario.

5.2 Arquitectura física

El primer paso a la hora de diseñar el sistema es conocer la arquitectura física que va a ofrecer soporte en todas sus operaciones. En la subsección 5.1 se mostró un esquema general en el que se identificaron los diferentes elementos que intervienen en el proceso de diseño conceptual del videojuego. Por un lado se encontraba el servidor, que proveerá determinados servicios según las demandas de la herramienta y, por otro, el gestor de bases de datos, que almacenará, modificará y extraerá de la base de datos asociada la información introducida por el usuario. Ambos elementos constituyen la arquitectura física del sistema, los cuales van a ser explicados con detalle a continuación.

5.2.1 Servidor de aplicaciones

La herramienta de autoría va a ser una aplicación JEE (Java Platform Enterprise Edition) y como tal necesita de un servidor de aplicaciones en el que instalarse y ejecutarse. Por esta razón, se hará uso de un servidor de aplicaciones WebSphere en su versión 7.0.

El servidor de aplicaciones WebSphere (WAS) [27] es uno de los productos más importantes de IBM. Está construido a partir de estándares abiertos como J2EE y servicios web. Actualmente cuenta con varias versiones disponibles, siendo la 8.5.5 la última, que fue publicada en el año 2013. WAS 7.0 cumple con la especificación Java EE 5 y es una versión estable, por lo cual es idónea para funcionar como servidor de la herramienta.

La herramienta se instalará en el servidor WAS y permanecerá en ejecución. El usuario, a través de su ordenador personal o de un dispositivo *tablet*, accederá a la aplicación mediante un navegador web.

5.2.2 Gestor de bases de datos

El otro elemento esencial de la arquitectura física del sistema es un gestor de bases de datos con el que manejar los datos de los diseños de videojuego. El gestor escogido para realizar las labores necesarias será MySQL en su versión 5.1.

MySQL [28] es un sistema gestor de bases de datos relacionales muy popular entre desarrolladores creado y proporcionado por MySQL AB. Consiste en un sistema cliente/servidor que se compone de un servidor SQL multihilo, varios programas clientes y bibliotecas, herramientas administrativas y una buena variedad de interfaces de programación. Hoy en día, este sistema continúa en constante mejora, aunque hay

disponibles versiones estables que facilitan la funcionalidad que se desea de este tipo de sistemas. Es el caso de la versión 5.1 escogida como gestor para la herramienta.

MySQL es un software utilizado en sitios webs populares como Twitter, Facebook e incluso Google, lo que ayuda a confiar en su efectividad. Además, es compatible con aplicaciones escritas en distintos lenguajes, como es el caso de Java. Su conectividad y velocidad en la lectura de datos lo hacen bastante apropiado para acceder a bases de datos vía Internet.

5.2.3 Esquema de la arquitectura

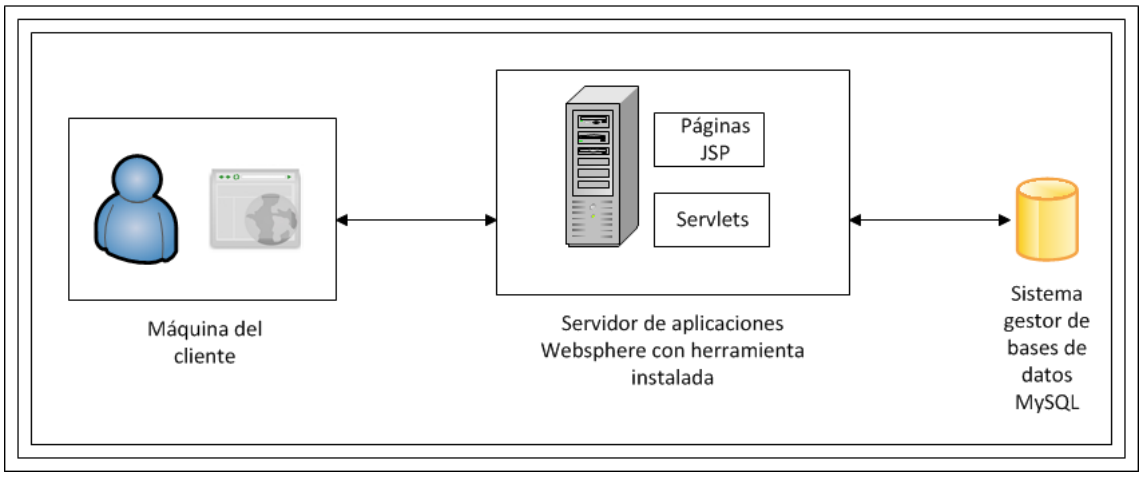


Ilustración 19. Esquema de la arquitectura física

5.3 Arquitectura software

En esta subsección se explicará primeramente el patrón de diseño escogido para el sistema. Una vez definido el patrón, se realizará un desglose de los diferentes componentes que conforman el sistema y que en su conjunto ofrecen la funcionalidad requerida.

5.3.1 Patrón de diseño

El patrón de diseño hace referencia a la arquitectura software más adecuada para el sistema. Al ser una aplicación web ofrecerá una interfaz gráfica con la que el usuario operará para diseñar su videojuego. Todas las acciones derivadas de la interacción herramienta de autoría-usuario serán gestionadas por el servidor de aplicaciones y la información generada será almacenada en una base de datos. Atendiendo a estos requerimientos la arquitectura software más apropiada es la que viene dada por el patrón Modelo-Vista-Controlador (MVC).

El MVC es un patrón de arquitectura software que separa la lógica de negocio, los datos y la interfaz de usuario en tres componentes diferentes: Modelo, Vista y Controlador.

- **Modelo**. Representa a los datos de la aplicación y es independiente del controlador y de la vista.

- **Vista**. Es la presentación de la información contenida en el Modelo de cara al usuario.

- **Controlador**. Responde a eventos (normalmente procedentes de la Vista) ejecutando la acción adecuada y crea el Modelo pertinente. Es el intermediario entre la Vista y el Modelo.

El flujo de control de este patrón es el siguiente:

1. El usuario efectúa una acción en la interfaz.

2. El Controlador trata el evento de entrada previamente registrado.

3. El Controlador notifica al Modelo la acción del usuario, lo que puede implicar un cambio en el estado del Modelo.

4. Se genera una nueva Vista que toma los datos del Modelo.

5. La interfaz de usuario espera otra interacción para comenzar de nuevo el proceso.

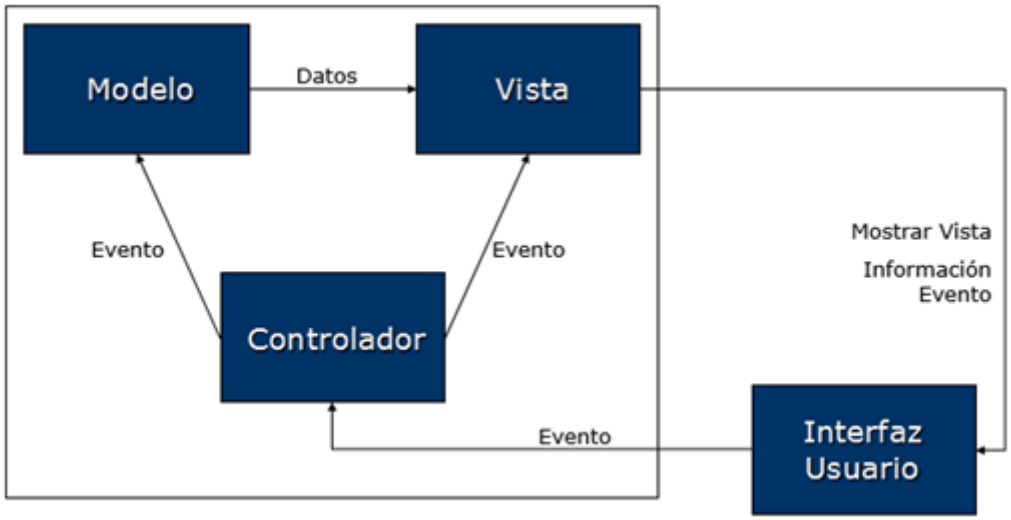


Ilustración 20. Patrón de diseño MVC

La separación en tres componentes bien diferenciados proporciona una gran flexibilidad en el desarrollo del sistema y buenas expectativas en cuanto a mantenimiento, ya que en caso de ser necesario algún tipo de corrección éste sólo se realizaría en un único lugar al evitar la mezcla de presentación e implementación. Esta arquitectura también posibilita la reutilización de sus componentes, por lo que es idónea su aplicación en el sistema.

5.3.2 Descripción de los subsistemas y sus componentes

En la arquitectura del sistema se van a distinguir tres subsistemas principales que se corresponden con las tres capas del patrón MVC: base de datos (Modelo), interfaz de usuario (Vista) y gestor de operaciones (Controlador). El siguiente diagrama ilustra cada uno de esos subsistemas junto con sus componentes y las relaciones que hay entre ellos:

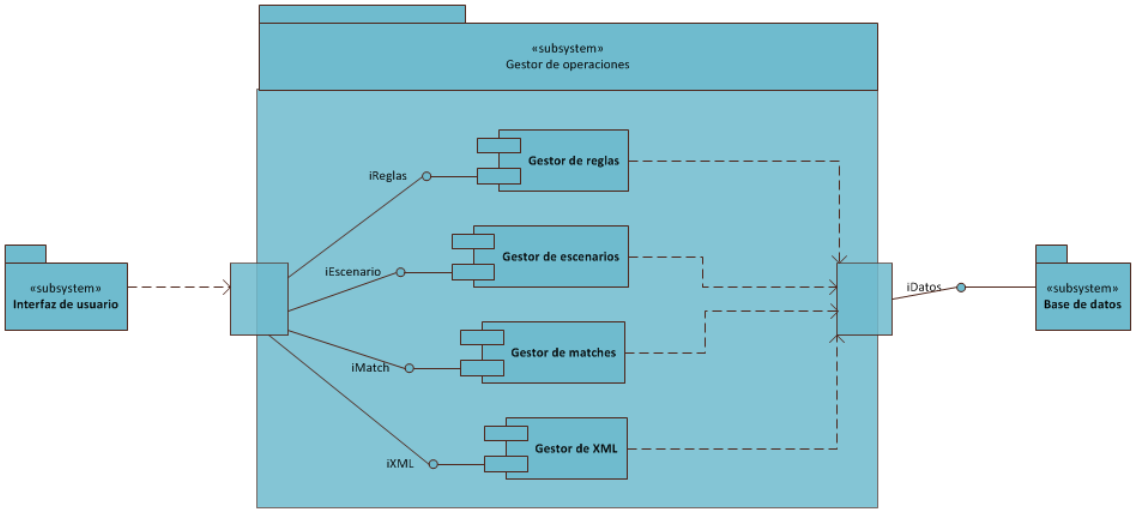


Ilustración 21. Diagrama de componentes

En los siguientes apartados se profundizará en los tres subsistemas y sus componentes y se realizará un seguimiento de ellos para ver si cubren todos los requisitos de software funcionales identificados en la sección de análisis.

5.3.2.1 Modelo de base de datos

La base de datos se encargará de gestionar la información necesaria para el diseño del videojuego. Las entidades JPA que representan cada una de las tablas de la base de datos, que serán detalladas más adelante, constituirán el Modelo dentro de la arquitectura MVC del sistema.

La información que maneja la base de datos está clasificada en tres grupos:

- Conjuntos de reglas, incluidos los elementos y subelementos que los forman.
- Escenarios, junto con sus elementos y subelementos.
- Matches.

La base de datos que va a almacenar todo esto será fiel a la estructura detallada en el modelo relacional que se presenta seguidamente:

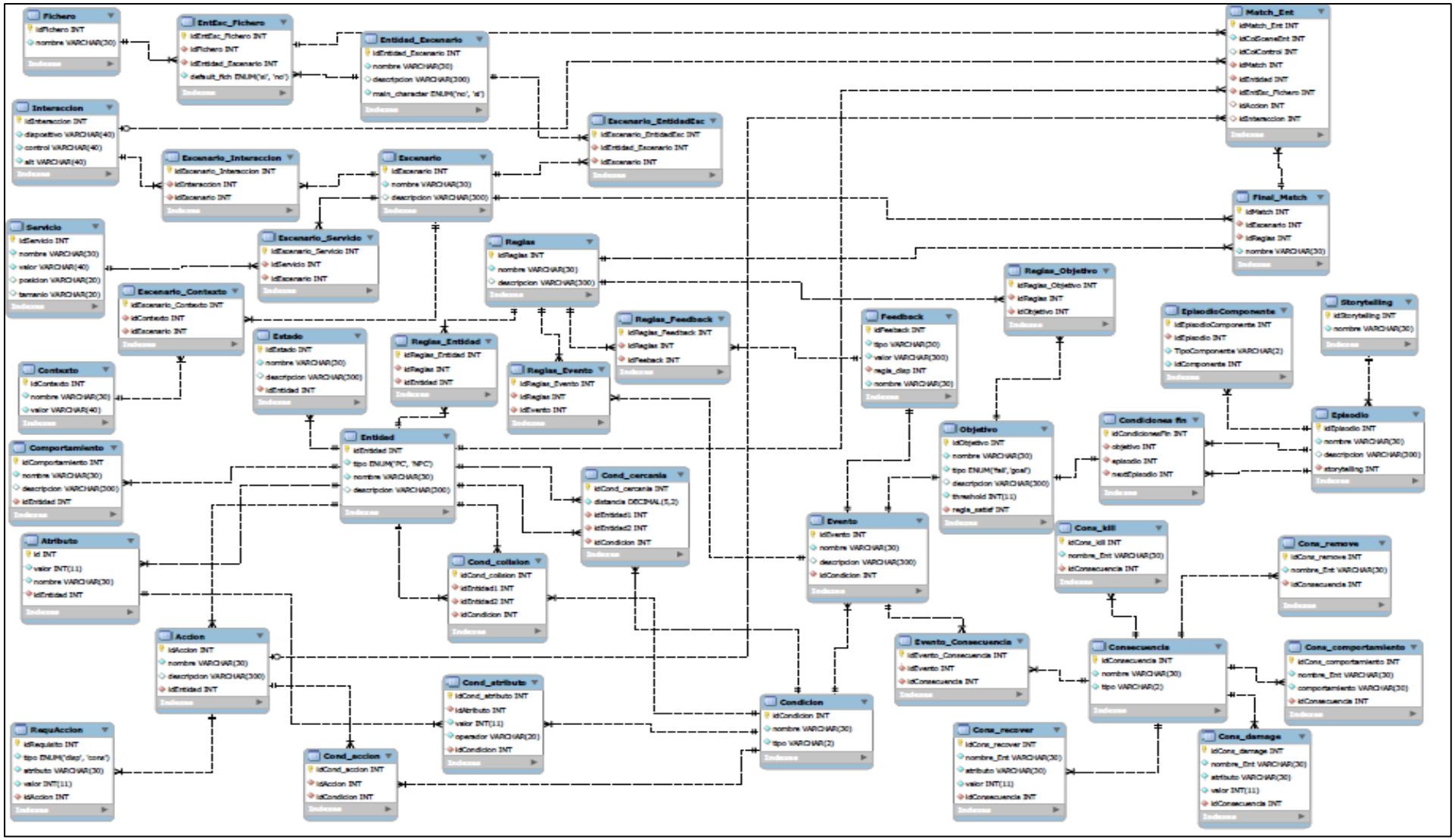


Ilustración 22. Modelo relacional de BBD

Como se puede apreciar en el modelo relacional, la base de datos contará con un variado número de tablas que contendrán los datos de los tres grupos de información disponibles. A continuación, el modelo relacional se va a explicar con detalle diferenciando las tablas que corresponden a los conjuntos de reglas, las que están asociadas a escenarios y las que forman parte de los matches.

5.3.2.1.1 Conjuntos de reglas

Las tablas de este grupo configuran la definición de cada conjunto de reglas.

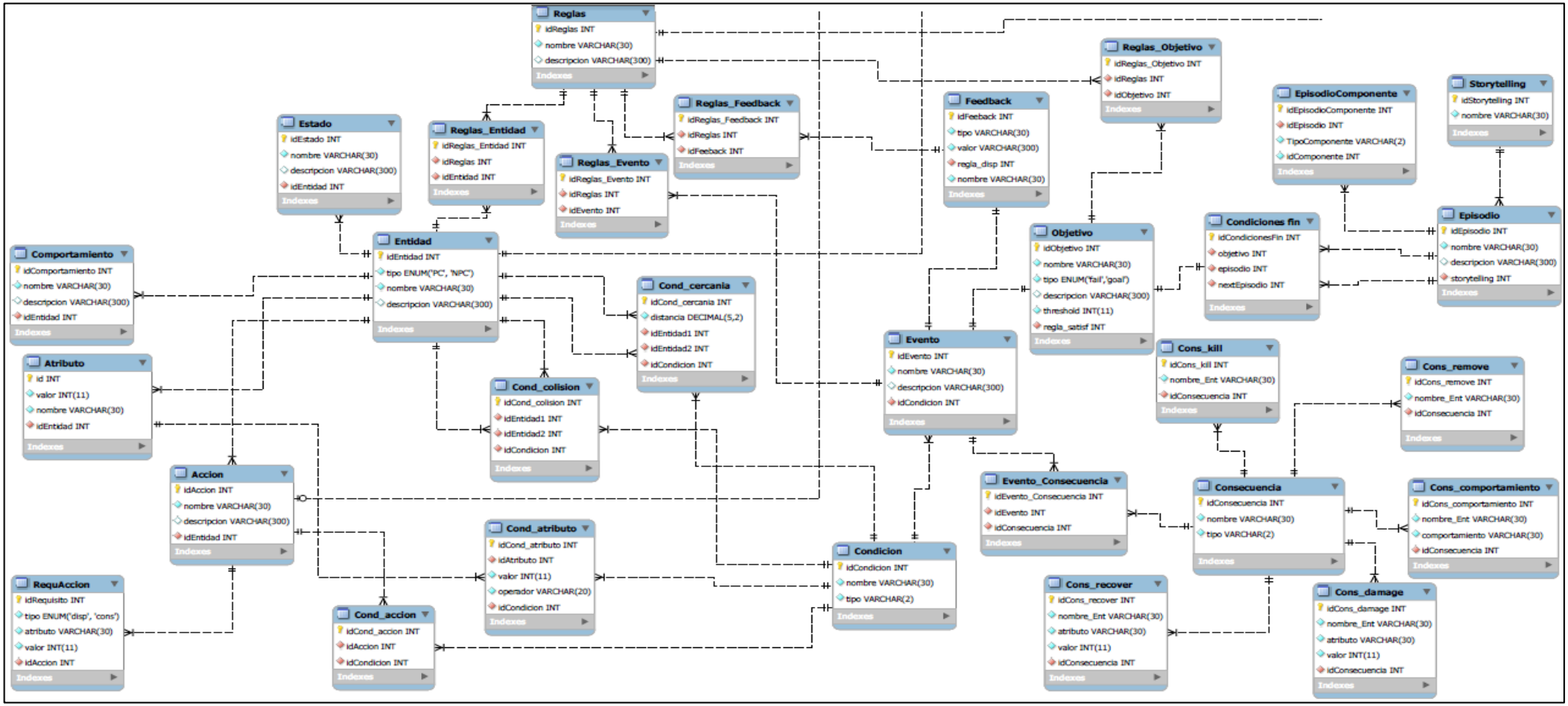


Ilustración 23. Tablas conjunto de reglas

5.3.2.1.1.1 Entidad

Representa una entidad de reglas. Contiene los campos básicos para su creación:

- **idEntidad**. Clave primaria de tipo entero que identifica de manera unívoca una entidad.
- **tipo**. Campo enumerado con los posibles tipos de la entidad. No puede tener valor nulo.
- **nombre**. Campo de texto con el nombre de la entidad. No puede tener valor nulo.
- **descripcion**. Campo de texto que contiene una descripción breve de la entidad.

5.3.2.1.1.2 Reglas_Entidad

Tabla auxiliar que hace referencia a una relación entre un conjunto de reglas y una entidad. Se compone de estos campos:

- **idReglas_Entidad**. Clave primaria de tipo entero que identifica de manera unívoca una tupla de esta tabla.
- **idReglas**. Clave ajena de tipo entero que identifica de manera unívoca un conjunto de reglas. No puede tener valor nulo.
- **idEntidad**. Clave ajena de tipo entero que identifica de manera unívoca una entidad de reglas. No puede tener valor nulo.

5.3.2.1.1.3 Estado

Representa un estado, que forma parte de la definición de una entidad de reglas. Consta de los siguientes campos:

- **idEstado**. Clave primaria de tipo entero que identifica de manera unívoca un estado.
- **nombre**. Campo de texto con el nombre del estado. No puede tener valor nulo.
- **descripcion**. Campo de texto que contiene una descripción breve del estado.
- **idEntidad**. Clave ajena de tipo entero que identifica de manera unívoca una entidad de reglas.

5.3.2.1.1.4 Comportamiento

Representa un comportamiento, que forma parte de la definición de una entidad de reglas. Consta de los siguientes campos:.

- **idComportamiento**. Clave primaria de tipo entero que identifica de manera unívoca un comportamiento.
- **nombre**. Campo de texto con el nombre del comportamiento. No puede tener valor nulo.
- **descripcion**. Campo de texto que contiene una descripción breve del comportamiento.
- **idEntidad**. Clave ajena de tipo entero que identifica de manera unívoca una entidad de reglas.

5.3.2.1.1.5 Atributo

Representa un atributo, que forma parte de la definición de una entidad de reglas. Consta de los siguientes campos:

- **id**. Clave primaria de tipo entero que identifica de manera unívoca un atributo.
- **valor**. Campo de tipo entero que almacena el valor del atributo. No puede tener valor nulo.
- **nombre**. Campo de texto con el nombre del atributo. No puede tener valor nulo.
- **idEntidad**. Clave ajena de tipo entero que identifica de manera unívoca una entidad de reglas.

5.3.2.1.1.6 Acción

Representa una acción, que forma parte de la definición de una entidad de reglas. Consta de los siguientes campos:

- **idAccion**. Clave primaria de tipo entero que identifica de manera unívoca una acción.
- **nombre**. Campo de texto con el nombre de la acción. No puede tener valor nulo.
- **descripcion**. Campo de texto que contiene una descripción breve de la acción.
- **idEntidad**. Clave ajena de tipo entero que identifica de manera unívoca una entidad de reglas.

5.3.2.1.1.7 RequAccion

Representa un requisito de acción, que forma parte de la definición de una acción. Consta de los siguientes campos:

- **idRequisito**. Clave primaria de tipo entero que identifica de manera unívoca un requisito de acción.
- **tipo**. Campo enumerado con los posibles tipos del requisito. No puede tener valor nulo.
- **atributo**. Campo de texto con el nombre del atributo al que hace referencia. No puede tener valor nulo.
- **valor**. Campo de tipo entero que almacena el valor del requisito. No puede tener valor nulo.
- **idAccion**. Clave ajena de tipo entero que identifica de manera unívoca una acción.

5.3.2.1.1.8 Evento

Representa un evento. Contiene los campos básicos para su creación:

- **idEvento**. Clave primaria de tipo entero que identifica de manera unívoca un evento.
- **nombre**. Campo de texto con el nombre del evento. No puede tener valor nulo.
- **descripcion**. Campo de texto que contiene una descripción breve de la acción.
- **idCondicion**. Clave ajena de tipo entero que identifica de manera unívoca una condición.

5.3.2.1.1.9 Reglas_Evento

Tabla auxiliar que hace referencia a una relación entre un conjunto de reglas y un evento. Se compone de estos campos:

- **idReglas_Evento**. Clave primaria de tipo entero que identifica de manera unívoca una tupla de esta tabla.
- **idReglas**. Clave ajena de tipo entero que identifica de manera unívoca un conjunto de reglas. No puede tener valor nulo.
- **idEvento**. Clave ajena de tipo entero que identifica de manera unívoca un evento. No puede tener valor nulo.

5.3.2.1.1.10 Condicion

Representa una condición, que forma parte de la definición de una acción. Consta de los siguientes campos:

- **idCondicion**. Clave primaria de tipo entero que identifica de manera unívoca una condición.
- **nombre**. Campo de texto con el nombre de la condición. No puede tener valor nulo.
- **tipo**. Campo de texto con las iniciales de los posibles tipos de condición. No puede tener valor nulo.

5.3.2.1.1.11 Cond_accion

Representa una condición de acción, que es un tipo de condición. Se compone de estos campos:

- **idCond_accion**. Clave primaria de tipo entero que identifica de manera unívoca una condición de acción.
- **idAccion**. Clave ajena de tipo entero que identifica de manera unívoca una acción.
- **idCondicion**. Clave ajena de tipo entero que identifica de manera unívoca una condición. No puede tener valor nulo.

5.3.2.1.1.12 Cond_atributo

Representa una condición de atributo, que es un tipo de condición. Se compone de estos campos:

- **idCond_atributo**. Clave primaria de tipo entero que identifica de manera unívoca una condición de atributo.
- **idAtributo**. Clave ajena de tipo entero que identifica de manera unívoca un atributo.
- **valor**. Campo de tipo entero que almacena el valor de la condición de atributo. No puede tener valor nulo.
- **operador**. Campo de texto que guarda el operador que acompaña al valor. No puede tener valor nulo.
- **idCondicion**. Clave ajena de tipo entero que identifica de manera unívoca una condición. No puede tener valor nulo.

5.3.2.1.1.13 Cond_colision

Representa una condición de colisión, que es un tipo de condición. Se compone de estos campos:

- **idCond_colision**. Clave primaria de tipo entero que identifica de manera unívoca una condición de colisión.
- **idEntidad1**. Clave ajena de tipo entero que identifica de manera unívoca una de las entidades que intervienen en este tipo de condición. No puede tener valor nulo.
- **idEntidad2**. Clave ajena de tipo entero que identifica de manera unívoca la otra entidad que interviene en este tipo de condición. No puede tener valor nulo.
- **idCondicion**. Clave ajena de tipo entero que identifica de manera unívoca una condición. No puede tener valor nulo.

5.3.2.1.1.14 Cond_cercania

Representa una condición de cercanía, que es un tipo de condición. Se compone de estos campos:

- **idCond_cercania**. Clave primaria de tipo entero que identifica de manera unívoca una condición de cercanía.
- **distancia**. Campo de tipo decimal que guarda la distancia entre las dos entidades que participan en este tipo de condición.
- **idEntidad1**. Clave ajena de tipo entero que identifica de manera unívoca una de las entidades que intervienen en este tipo de condición. No puede tener valor nulo.
- **idEntidad2**. Clave ajena de tipo entero que identifica de manera unívoca la otra entidad que interviene en este tipo de condición. No puede tener valor nulo.
- **idCondicion**. Clave ajena de tipo entero que identifica de manera unívoca una condición. No puede tener valor nulo.

5.3.2.1.1.15 Evento_Consecuencia

Tabla auxiliar que hace referencia a una relación entre un evento y una consecuencia. Se compone de estos campos:

- **idEvento_Consecuencia**. Clave primaria de tipo entero que identifica de manera unívoca una tupla de esta tabla.

- **idEvento**. Clave ajena de tipo entero que identifica de manera unívoca un evento. No puede tener valor nulo.

- **idConsecuencia**. Clave ajena de tipo entero que identifica de manera unívoca una consecuencia. No puede tener valor nulo.

5.3.2.1.1.16 Consecuencia

Representa una consecuencia, que forma parte de la definición de una acción. Consta de los siguientes campos:

- **idConsecuencia**. Clave primaria de tipo entero que identifica de manera unívoca una consecuencia.

- **nombre**. Campo de texto con el nombre de la consecuencia. No puede tener valor nulo.

- **tipo**. Campo de texto con las iniciales de los posibles tipos de consecuencia. No puede tener valor nulo.

5.3.2.1.1.17 Cons_recover

Representa una consecuencia de tipo "recuperación". Se compone de estos campos:

- **idCons_recover**. Clave primaria de tipo entero que identifica de manera unívoca una consecuencia de tipo "recuperación".

- **nombre_Ent**. Campo de texto con el nombre de la entidad implicada en la consecuencia. No puede tener valor nulo.

- **atributo**. Campo de texto que indica el atributo de la entidad que se ve afectado por la consecuencia. No puede tener valor nulo.

- **valor**. Campo de texto de tipo entero que almacenará la cantidad que se le sumará al valor del atributo al producirse este evento. No puede tener valor nulo.

- **idConsecuencia**. Clave ajena de tipo entero que identifica de manera unívoca una consecuencia. No puede tener valor nulo.

5.3.2.1.1.18 Cons_damage

Representa una consecuencia de tipo "daño". Se compone de estos campos:

- **idCons_damage**. Clave primaria de tipo entero que identifica de manera unívoca una consecuencia de tipo "daño".

- **nombre_Ent**. Campo de texto con el nombre de la entidad implicada en la consecuencia. No puede tener valor nulo.

- **atributo**. Campo de texto que indica el atributo de la entidad que se ve afectado por la consecuencia. No puede tener valor nulo.

- **valor**. Campo de texto de tipo entero que almacenará la cantidad que se le restará al valor del atributo al producirse este evento. No puede tener valor nulo.

- **idConsecuencia**. Clave ajena de tipo entero que identifica de manera unívoca una consecuencia. No puede tener valor nulo.

5.3.2.1.1.19 Cons_comportamiento

Representa una consecuencia de tipo "comportamiento". Se compone de estos campos:

- **idCons_comportamiento**. Clave primaria de tipo entero que identifica de manera unívoca una consecuencia de tipo "comportamiento".

- **nombre_Ent**. Campo de texto con el nombre de la entidad implicada en la consecuencia. No puede tener valor nulo.

- **comportamiento**. Campo de texto que indica el comportamiento de la entidad que se ve afectado por la consecuencia. No puede tener valor nulo.

- **idConsecuencia**. Clave ajena de tipo entero que identifica de manera unívoca una consecuencia. No puede tener valor nulo.

5.3.2.1.1.20 Cons_remove

Representa una consecuencia de tipo "borrado". Se compone de estos campos:

- **idCons_remove**. Clave primaria de tipo entero que identifica de manera unívoca una consecuencia de tipo "borrado".

- **nombre_Ent**. Campo de texto con el nombre de la entidad implicada en la consecuencia. No puede tener valor nulo.

- **idConsecuencia**. Clave ajena de tipo entero que identifica de manera unívoca una consecuencia. No puede tener valor nulo.

5.3.2.1.1.21 Cons_kill

Representa una consecuencia de tipo "muerte". Se compone de estos campos:

- **idCons_kill**. Clave primaria de tipo entero que identifica de manera unívoca una consecuencia de tipo "muerte".

- **nombre_Ent**. Campo de texto con el nombre de la entidad implicada en la consecuencia. No puede tener valor nulo.

- **idConsecuencia**. Clave ajena de tipo entero que identifica de manera unívoca una consecuencia. No puede tener valor nulo.

5.3.2.1.1.22 Objetivo

Representa un objetivo. Contiene los campos básicos para su creación:

- **idObjetivo**. Clave primaria de tipo entero que identifica de manera unívoca un objetivo.

- **nombre**. Campo de texto con el nombre del objetivo. No puede tener valor nulo.

- **tipo**. Campo enumerado con los posibles tipos del objetivo. No puede tener valor nulo.

- **descripcion**. Campo de texto que contiene una descripción breve del objetivo.

- **threshold**. Campo de tipo entero que guarda el valor del threshold del objetivo. No puede tener valor nulo.

- **regla_satisf**. Clave ajena de tipo entero que identifica de manera unívoca un evento. Dicho evento es el encargado de informar que el jugador ha cumplido el objetivo.

5.3.2.1.1.23 Reglas_Objetivo

Tabla auxiliar que hace referencia a una relación entre un conjunto de reglas y un objetivo. Se compone de estos campos:

- **idReglas_Objetivo**. Clave primaria de tipo entero que identifica de manera unívoca una tupla de esta tabla.

- **idReglas**. Clave ajena de tipo entero que identifica de manera unívoca un conjunto de reglas. No puede tener valor nulo.

- **idObjetivo**. Clave ajena de tipo entero que identifica de manera unívoca un objetivo. No puede tener valor nulo.

5.3.2.1.1.24 Feedback

Representa un feedback. Contiene los campos básicos para su creación:

- **idFeedback**. Clave primaria de tipo entero que identifica de manera unívoca un feedback.

- **tipo**. Campo de texto que guarda el tipo de feedback. No puede tener valor nulo.

- **valor**. Campo de texto que almacena el valor del feedback. No puede tener valor nulo.

- **regla_disp**. Clave ajena de tipo entero que identifica de manera unívoca un evento. Dicho evento es el encargado de disparar el feedback para que muestre su valor (un mensaje informativo) en la interfaz del videojuego. No puede tener valor nulo.

- **nombre**. Campo de texto con el nombre del feedback. No puede tener valor nulo.

5.3.2.1.1.25 Reglas_Feedback

Tabla auxiliar que hace referencia a una relación entre un conjunto de reglas y un feedback. Se compone de estos campos:

- **idReglas_Feedback**. Clave primaria de tipo entero que identifica de manera unívoca una tupla de esta tabla.

- **idReglas**. Clave ajena de tipo entero que identifica de manera unívoca un conjunto de reglas. No puede tener valor nulo.

- **idFeedback**. Clave ajena de tipo entero que identifica de manera unívoca un feedback. No puede tener valor nulo.

5.3.2.1.1.26 Storytelling

Representa un storytelling. Contiene los campos básicos para su creación:

- **idStorytelling**. Clave primaria de tipo entero que identifica de manera unívoca un storytelling.

- **nombre**. Campo de texto con el nombre del storytelling. No puede tener valor nulo.

5.3.2.1.1.27 Episodio

Representa un episodio, que forma parte de la definición de un storytelling. Consta de los siguientes campos:

- **idEpisodio**. Clave primaria de tipo entero que identifica de manera unívoca un episodio.

- **nombre**. Campo de texto con el nombre del episodio. No puede tener valor nulo.

- **descripcion**. Campo de texto que contiene una descripción breve del episodio.

- **storytelling**. Clave ajena de tipo entero que identifica de manera unívoca un storytelling. No puede tener valor nulo.

5.3.2.1.1.28 EpisodioComponente

Hace referencia a un componente del episodio, que puede ser una entidad, un objetivo, etc. Dispone de estos campos:

- **idEpisodioComponente**. Clave primaria de tipo entero que identifica de manera unívoca un componente de un episodio.

- **idEpisodio**. Clave ajena de tipo entero que identifica de manera unívoca un episodio. No puede tener valor nulo.

- **TipoComponente**. Campo de texto con las iniciales de los posibles tipos de componente. No puede tener valor nulo.

- **idComponente**. Clave ajena de tipo entero que identifica de manera unívoca un componente. No puede tener valor nulo.

5.3.2.1.1.29 Condiciones_fin

Representa una condición de fin, que forma parte de la definición de un episodio. La condición de fin indica el objetivo que ha de alcanzarse para dar por finalizado un episodio de un videojuego y dar inicio al siguiente. Consta de los siguientes campos:

- **idCondicionesFin**. Clave primaria de tipo entero que identifica de manera unívoca una condición de fin.

- **objetivo**. Clave ajena de tipo entero que identifica de manera unívoca un objetivo. No puede tener valor nulo.

- **episodio**. Clave ajena de tipo entero que identifica de manera unívoca un episodio. No puede tener valor nulo.

- **nextEpisodio**. Clave ajena de tipo entero que identifica de manera unívoca el siguiente episodio. No puede tener valor nulo.

5.3.2.1.1.30 Reglas

Representa un conjunto de reglas. Contiene los campos básicos para su creación:

- **idReglas**. Clave primaria de tipo entero que identifica de manera unívoca un conjunto de reglas.

- **nombre**. Campo de texto con el nombre del conjunto de reglas. No puede tener valor nulo.

- **descripcion**. Campo de texto que contiene una descripción breve del conjunto de reglas.

5.3.2.1.2 Escenarios

Las tablas de este grupo configuran la definición de cada escenario.

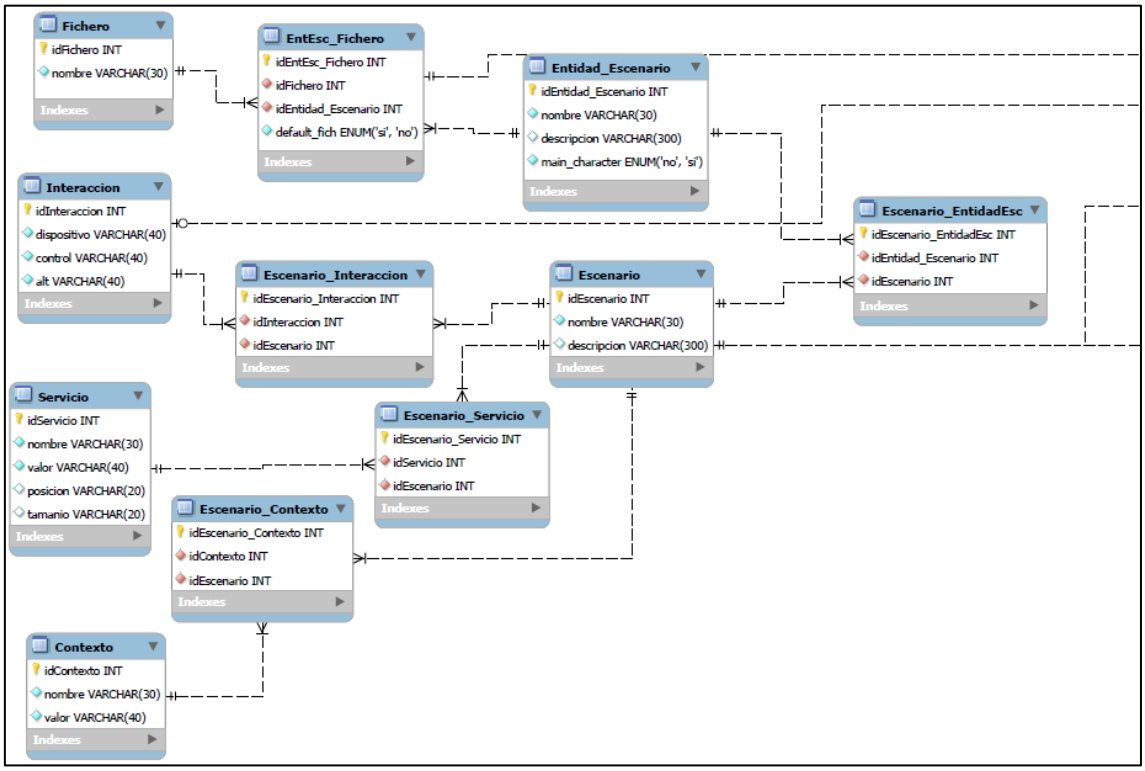


Ilustración 24. Tablas escenarios

5.3.2.1.2.1 Entidad_Escenario

Representa una entidad de escenario. Contiene los campos básicos para su creación:

- **idEntidad_Escenario**. Clave primaria de tipo entero que identifica de manera unívoca una entidad de escenario.

- **nombre**. Campo de texto con el nombre de la entidad. No puede tener valor nulo.

- **descripcion**. Campo de texto que contiene una descripción breve de la entidad.

- **main_character**. Campo enumerado que indica si la entidad hace referencia a un personaje controlado por el jugador o no.

5.3.2.1.2.2 Escenario_EntidadEsc

Tabla auxiliar que hace referencia a una relación entre un escenario y una entidad de escenario. Se compone de estos campos:

- **idEscenario_EntidadEsc**. Clave primaria de tipo entero que identifica de manera unívoca una tupla de esta tabla.

- **idEntidad_Escenario**. Clave ajena de tipo entero que identifica de manera unívoca una entidad de escenario. No puede tener valor nulo.

- **idEscenario**. Clave ajena de tipo entero que identifica de manera unívoca un escenario. No puede tener valor nulo.

5.3.2.1.2.3 EntEsc_Fichero

Tabla auxiliar que hace referencia a una relación entre una entidad de escenario y un fichero. Se compone de estos campos:

- **idEntEsc_Fichero**. Clave primaria de tipo entero que identifica de manera unívoca una tupla de esta tabla.

- **idFichero**. Clave ajena de tipo entero que identifica de manera unívoca un fichero. No puede tener valor nulo.

- **idEntidad_Escenario**. Clave ajena de tipo entero que identifica de manera unívoca una entidad de escenario. No puede tener valor nulo.

- **default_fich**. Campo enumerado que especifica si la entidad de escenario contiene un fichero por defecto o no.

5.3.2.1.2.4 Fichero

Representa un fichero, que forma parte de la definición de una entidad de reglas. El fichero al que hace referencia contiene una posible animación de la entidad de escenario. Consta de los siguientes campos:

- **idFichero**. Clave primaria de tipo entero que identifica de manera unívoca un fichero.

- **nombre**. Campo de texto con el nombre del fichero. No puede tener valor nulo.

5.3.2.1.2.5 Interaccion

Representa una interacción. Contiene los campos básicos para su creación:

- **idInteraccion**. Clave primaria de tipo entero que identifica de manera unívoca una interacción.
- **dispositivo**. Campo de texto que almacena el dispositivo hardware que permitirá la interacción como, por ejemplo, un teclado.
- **control**. Campo de texto que indica la manera en que se produce la interacción (pulsando un botón, por ejemplo).
- **alt**. Campo de texto que guarda un control alternativo.

5.3.2.1.2.6 Escenario_Interaccion

Tabla auxiliar que hace referencia a una relación entre un escenario y una interacción. Se compone de estos campos:

- **idEscenario_Interaccion**. Clave primaria de tipo entero que identifica de manera unívoca una tupla de esta tabla.
- **idInteraccion**. Clave ajena de tipo entero que identifica de manera unívoca una interacción. No puede tener valor nulo.
- **idEscenario**. Clave ajena de tipo entero que identifica de manera unívoca un escenario. No puede tener valor nulo.

5.3.2.1.2.7 Servicio

Representa un servicio. Contiene los campos básicos para su creación:

- **idServicio**. Clave primaria de tipo entero que identifica de manera unívoca un servicio.
- **nombre**. Campo de texto con el nombre del servicio. No puede tener valor nulo.
- **valor**. Campo de tipo entero que almacena el valor del servicio. No puede tener valor nulo.
- **posicion**. Campo de texto que guarda la posición en la pantalla del videojuego del servicio.
- **tamanio**. Campo de texto que guarda el tamaño del servicio dentro de la pantalla del videojuego.

5.3.2.1.2.8 Escenario_Servicio

Tabla auxiliar que hace referencia a una relación entre un escenario y un servicio. Se compone de estos campos:

- **idEscenario_Servicio**. Clave primaria de tipo entero que identifica de manera unívoca una tupla de esta tabla.
- **idServicio**. Clave ajena de tipo entero que identifica de manera unívoca un servicio. No puede tener valor nulo.
- **idEscenario**. Clave ajena de tipo entero que identifica de manera unívoca un escenario. No puede tener valor nulo.

5.3.2.1.2.9 Contexto

Representa un elemento de contexto. Contiene los campos básicos para su creación:

- **idContexto**. Clave primaria de tipo entero que identifica de manera unívoca un elemento de contexto.
- **nombre**. Campo de texto con el nombre del elemento de contexto. No puede tener valor nulo.
- **valor**. Campo de tipo entero que almacena el valor del elemento de contexto. No puede tener valor nulo.

5.3.2.1.2.10 Escenario_Contexto

Tabla auxiliar que hace referencia a una relación entre un escenario y un elemento de contexto. Se compone de estos campos:

- **idEscenario_Contexto**. Clave primaria de tipo entero que identifica de manera unívoca una tupla de esta tabla.
- **idContexto**. Clave ajena de tipo entero que identifica de manera unívoca un elemento de contexto. No puede tener valor nulo.
- **idEscenario**. Clave ajena de tipo entero que identifica de manera unívoca un escenario. No puede tener valor nulo.

5.3.2.1.2.11 Escenario

Representa un escenario. Contiene los campos básicos para su creación:

- **idEscenario**. Clave primaria de tipo entero que identifica de manera unívoca un escenario.

- **nombre**. Campo de texto con el nombre del escenario. No puede tener valor nulo.

- **descripcion**. Campo de texto que contiene una descripción breve del escenario.

5.3.2.1.3 Matches

Las tablas de este grupo configuran la definición de cada match o unión entre conjunto de reglas y escenario.

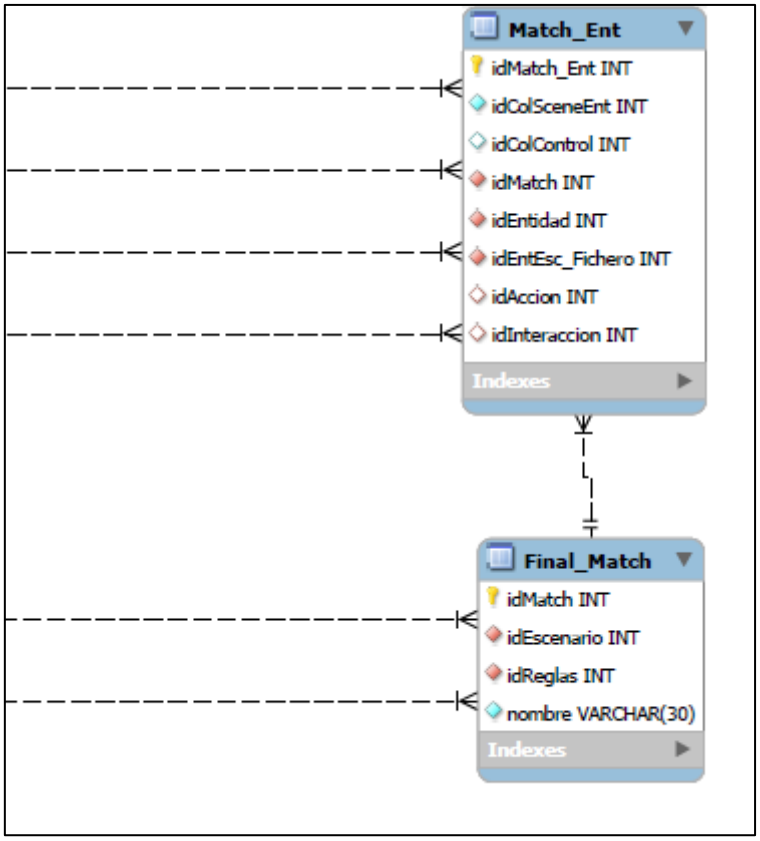


Ilustración 25. Tablas matches

5.3.2.1.3.1 Final_Match

Representa un match. Contiene los campos básicos para su creación:

- **idMatch**. Clave primaria de tipo entero que identifica de manera unívoca un match.

- **idEscenario**. Clave ajena de tipo entero que identifica de manera unívoca un escenario.

- **idReglas**. Clave ajena de tipo entero que identifica de manera unívoca un conjunto de reglas.

- **nombre**. Campo de texto con el nombre del match. No puede tener valor nulo.

5.3.2.1.3.2 Match_Ent

Es el elemento principal del match. Representa la unión de los elementos del conjunto de reglas y del escenario.

- **idMatch_Ent**. Clave primaria de tipo entero que identifica de manera unívoca a un elemento de match.
- **idColSceneEnt**. Campo de tipo entero que identifica el número de fila de una entidad de escenario en la pantalla de match. No puede tener valor nulo.
- **idColControl**. Campo de tipo entero que identifica el número de fila de un control en la pantalla de match.
- **idMatch**. Clave ajena de tipo entero que identifica de manera unívoca un match. No puede tener valor nulo.
- **idEntidad**. Clave ajena de tipo entero que identifica de manera unívoca una entidad de reglas.
- **idEntEsc_Fichero**. Clave ajena de tipo entero que identifica de manera unívoca una relación entidad de escenario-fichero.
- **idAccion**. Clave ajena de tipo entero que identifica de manera unívoca una acción.
- **idInteraccion**. Clave ajena de tipo entero que identifica de manera unívoca una interacción.

5.3.2.2 Interfaz de usuario

La Vista de la arquitectura MVC del sistema será la interfaz de usuario. A través de ella, el docente interaccionará con el Controlador, que accederá al Modelo para crear, modificar o consultar datos y proporcionárselos al usuario en formato de presentación.

Debido a la importancia de este subsistema, a continuación se va a mostrar una serie de prototipos de las pantallas que ilustran un posible aspecto de la interfaz de usuario previo a su desarrollo. De este modo, el cliente podrá hacerse una idea del futuro aspecto de la herramienta y dar el visto bueno o aportar nuevas ideas para su mejora.

Los prototipos se han realizado con el software Axure RP [29] en su versión libre. Debido a la buena variedad de librerías de componentes que ofrece y su facilidad de uso, se ha optado por crear unos prototipos más cercanos a una aplicación web real que un diseño a bajo nivel en papel u otros medios.

5.3.2.2.1 Página de inicio

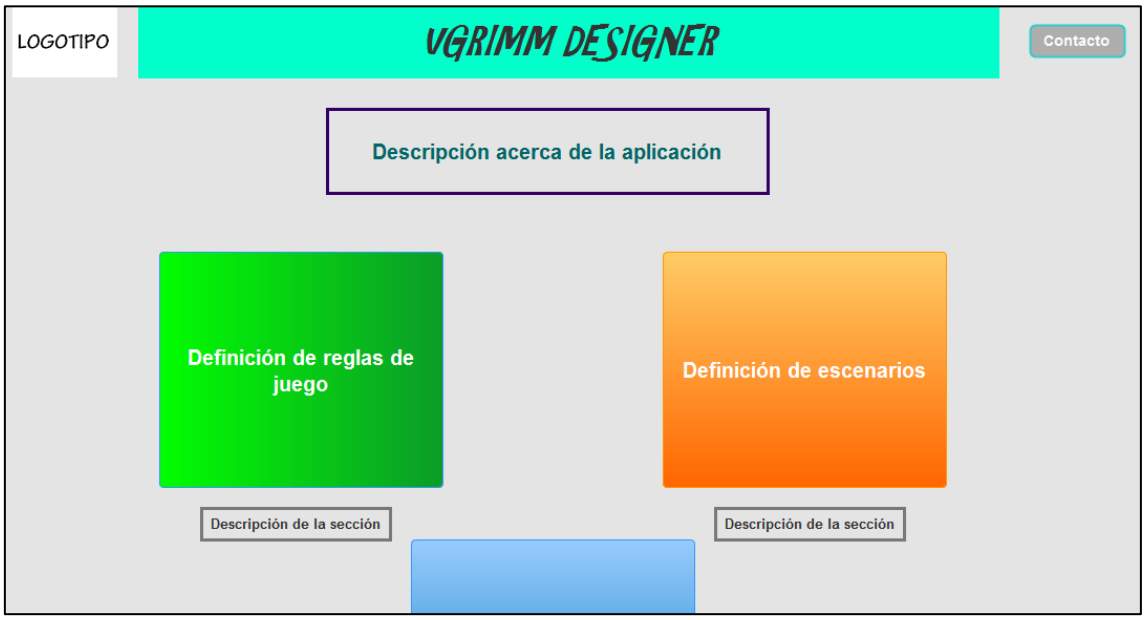


Ilustración 26. Pantalla de inicio

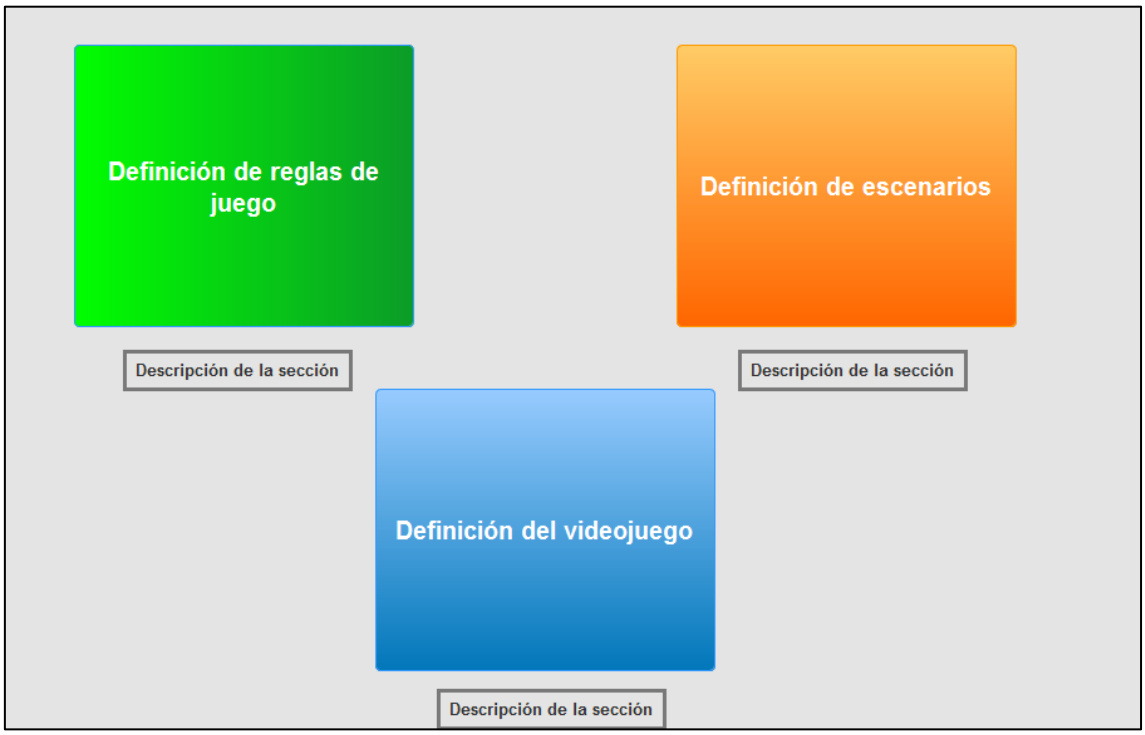


Ilustración 27. Pantalla de inicio (2)

De acuerdo con las ilustraciones, cuando el usuario accede a la aplicación a través de la Web se muestra la página de inicio. En la parte superior de ella se encuentran a modo de cabecera el logotipo distintivo de la herramienta, su nombre (*VGrimm Designer*) y un botón con el texto "Contacto". Al pulsar sobre este botón aparece una ventana modal que resume el cometido de la herramienta. Esta cabecera es común en todas las páginas. Debajo del título es posible leer una breve descripción de la aplicación.

Inmediatamente después de la cabecera, se exponen tres grandes botones que permiten acceder a las secciones principales de la herramienta. Bajo cada uno de ellos se puede leer una pequeña descripción acerca de dichas secciones.

5.3.2.2.2 Pantalla de presentación de una sección

Ilustración 28. Pantalla de presentación de la sección de reglas

La ilustración 23 hace referencia a la pantalla que se muestra inmediatamente después de pulsar uno de los botones de la página de inicio. Es similar en las tres secciones y en ella se ofrece una explicación más detallada de la sección seleccionada y un botón de acceso a la página inicial de dicha sección. Se permite al usuario volver a la página de inicio a través de un mecanismo de "migas de pan" que facilita la navegación y la orientación.

5.3.2.2.3 Ventana modal general de búsqueda y agregación de elementos

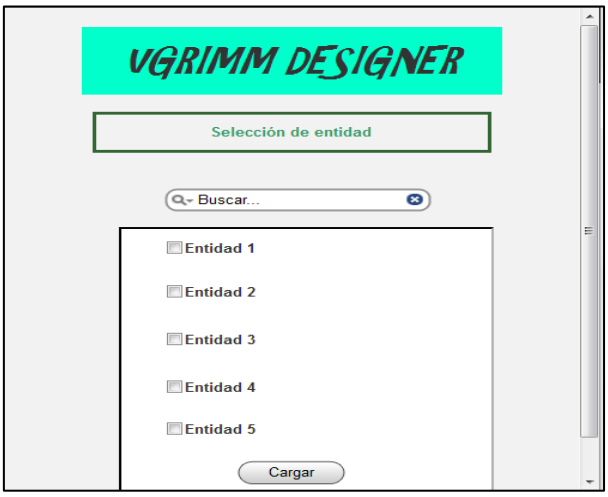


Ilustración 29. Ventana modal de búsqueda y agregación

En la ilustración 24 se presenta la ventana modal tipo que ofrece los servicios de búsqueda y agregación. Esta ventana es común a todos los elementos, ya sean conjuntos de reglas, escenarios, matches, elementos o subelementos. Justo debajo del título de la ventana se puede ver un pequeño campo de búsqueda que permite localizar elementos a través de su nombre. A continuación, se listan todos los elementos almacenados o, en consecuencia, los resultados de una búsqueda efectuada.

A la hora de la agregación, ésta se puede realizar de dos maneras distintas. En el caso de que se quiera cargar un determinado elemento (entidad, evento, etc.) para consultar o editar sus datos, se permite la selección de una única opción a través de *radio buttons*. La otra posibilidad consiste en la agregación de determinados subelementos a un elemento (por ejemplo, una acción a una entidad de reglas), por lo que se habilita la selección múltiple por medio de *check boxes*. Sea como fuere, la agregación o la carga sólo tienen lugar en el momento en que se pulsa el botón de cargar.

5.3.2.2.4 Pantalla de definición de entidades de reglas

Ilustración 30. Pantalla de entidades de reglas

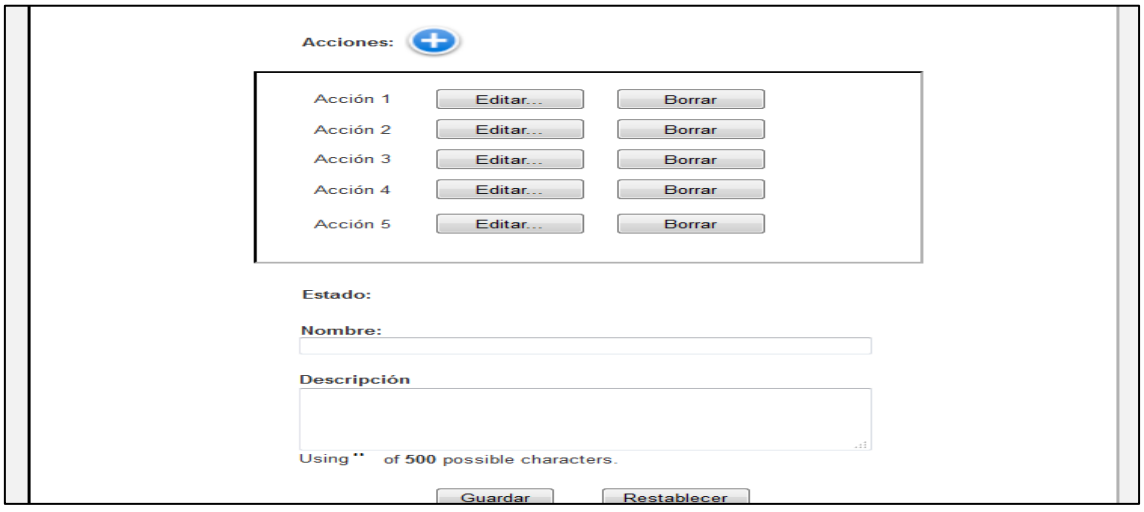


Ilustración 31. Pantalla de entidades de reglas (2)

Las ilustraciones 25 y 26 se corresponden con la pantalla de creación o modificación de entidades de reglas. El mecanismo de "migas de pan" se sigue mostrando debajo de la cabecera del mismo modo que en la pantalla de presentación de secciones. En el lado izquierdo de la página se muestra un menú vertical con todo el recorrido que debería realizarse para obtener un conjunto de reglas, desde el nivel 1 hasta el nivel 4. Si el usuario pulsa cualquiera de las opciones se le lleva de inmediato a la página de creación del elemento seleccionado. Este menú se adaptará automáticamente a la resolución del dispositivo desde el que se acceda a la aplicación.

En el lado derecho se sitúa el formulario principal de creación o modificación de la entidad. Encima de éste es posible leer unas pequeñas instrucciones que indican cómo rellenar sus campos. En la parte superior del formulario se encuentra un botón que al pulsarlo muestra la ventana modal de carga de una entidad. Los campos del formulario son los siguientes:

- **Nombre**. Campo de texto obligatorio en el que se introduce el nombre de la entidad.
- **Tipo**. *Combo box* que permite la selección del tipo de entidad.
- **Descripción**. Área de texto de rellenado opcional en el que se introduce una breve descripción de la entidad.
- **Atributos y acciones**. Áreas en las que se sitúan los atributos o acciones agregados a la entidad. Se da la posibilidad de borrar de la entidad los elementos que se deseen o crear otros nuevos.
- **Estado o comportamiento**. Dependiendo del tipo escogido, será preciso rellenar los datos del estado o del comportamiento de la entidad.

En último lugar, se añaden dos botones, uno para enviar la información introducida al controlador correspondiente y tratarla, y otro para limpiar el formulario.

5.3.2.2.5 Ventana modal de creación de atributos

Ilustración 32. Ventana modal de creación de atributos

Como su propio nombre indica, en ella se adjunta el formulario correspondiente para la creación y guardado de un atributo. Éste se compone de dos campos fundamentales:

- **Nombre atributo**. Campo de texto obligatorio en el que se introduce el nombre del atributo.

- **Valor**. Campo de texto obligatorio en el que se incluye el valor que tendrá el atributo creado.

5.3.2.2.6 Ventana modal de creación de acciones

Ilustración 33. Ventana modal de creación de acciones

La ilustración 28 se corresponde con el aspecto que tendría la ventana modal con el formulario de creación de acciones, el cual se compone de lo siguiente:

- **Nombre acción**. Campo de texto obligatorio en el que se introduce el nombre de la acción.

- **Descripción**. Área de texto opcional en la que se incluye una breve descripción de la acción.

- **Requisitos**. Área en la cual se listan los requisitos agregados a la acción. Se pueden crear requisitos nuevos, agregar otros o borrar los que ya estén incluidos.

5.3.2.2.7 Ventana modal de creación de requisitos

Ilustración 34. Ventana modal de creación de requisitos

La ilustración 29 hace referencia a la ventana modal que contendrá el formulario de creación de requisitos. Este formulario viene incluido con los siguientes campos, todos ellos de obligada cumplimentación:

- **Nombre atributo**. Campo de texto obligatorio en el que se introduce el nombre del atributo que se necesita para ejecutar la acción.
- **Tipo**. *Combo Box* que permite la selección del tipo de requisito.
- **Valor necesario**. Campo de texto en el que se introduce el valor de atributo necesario para llevar a cabo la acción.

5.3.2.2.8 Pantalla de definición de eventos

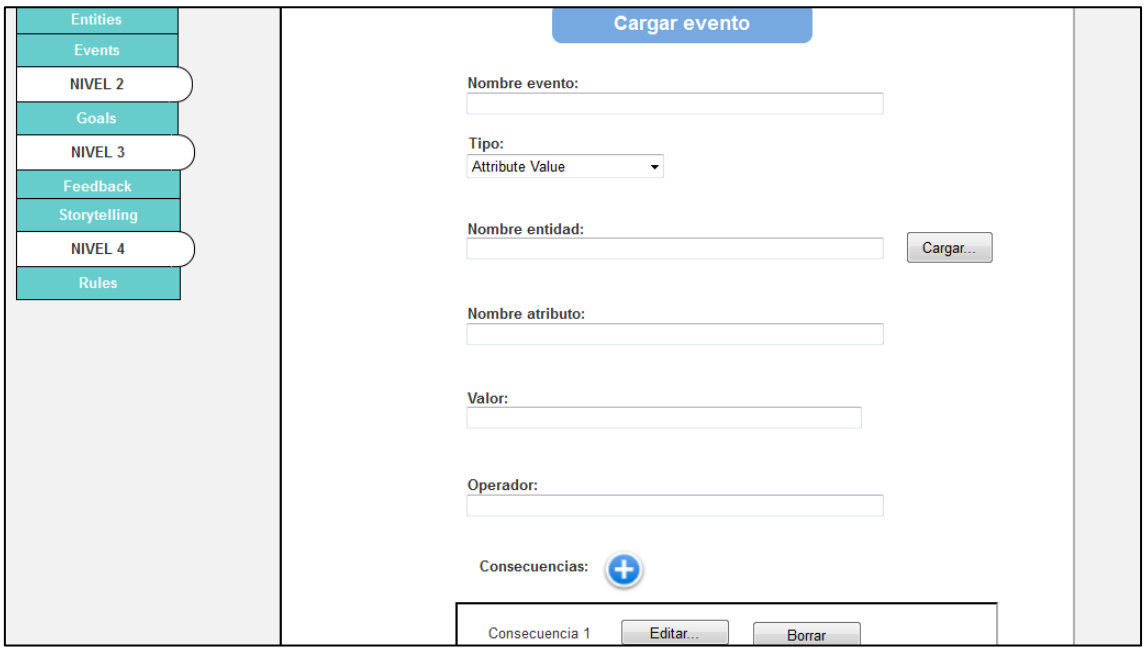


Ilustración 35. Pantalla de eventos

En esta pantalla el usuario puede crear o editar eventos. Presenta un aspecto similar al de definición de entidades de reglas, ya que en el lado izquierdo aparece el mismo menú vertical y en el derecho el formulario con los campos a rellenar con los datos del evento. La diferencia radica en los propios campos:

- **Nombre evento**. Campo de texto obligatorio en el que se introduce el nombre del evento.
- **Tipo**. *Combo Box* que permite la selección del tipo de evento.
- **Campos de condición**. Según el tipo de evento escogido, se mostrarán unos campos u otros relativos a la condición que hace posible la ejecución del evento.
- **Consecuencias**. Área en la cual se listan las consecuencias agregadas al evento. Se pueden crear consecuencias nuevas, agregar otras o borrar las que ya estén incluidas.

5.3.2.2.9 Ventana modal de creación de consecuencias

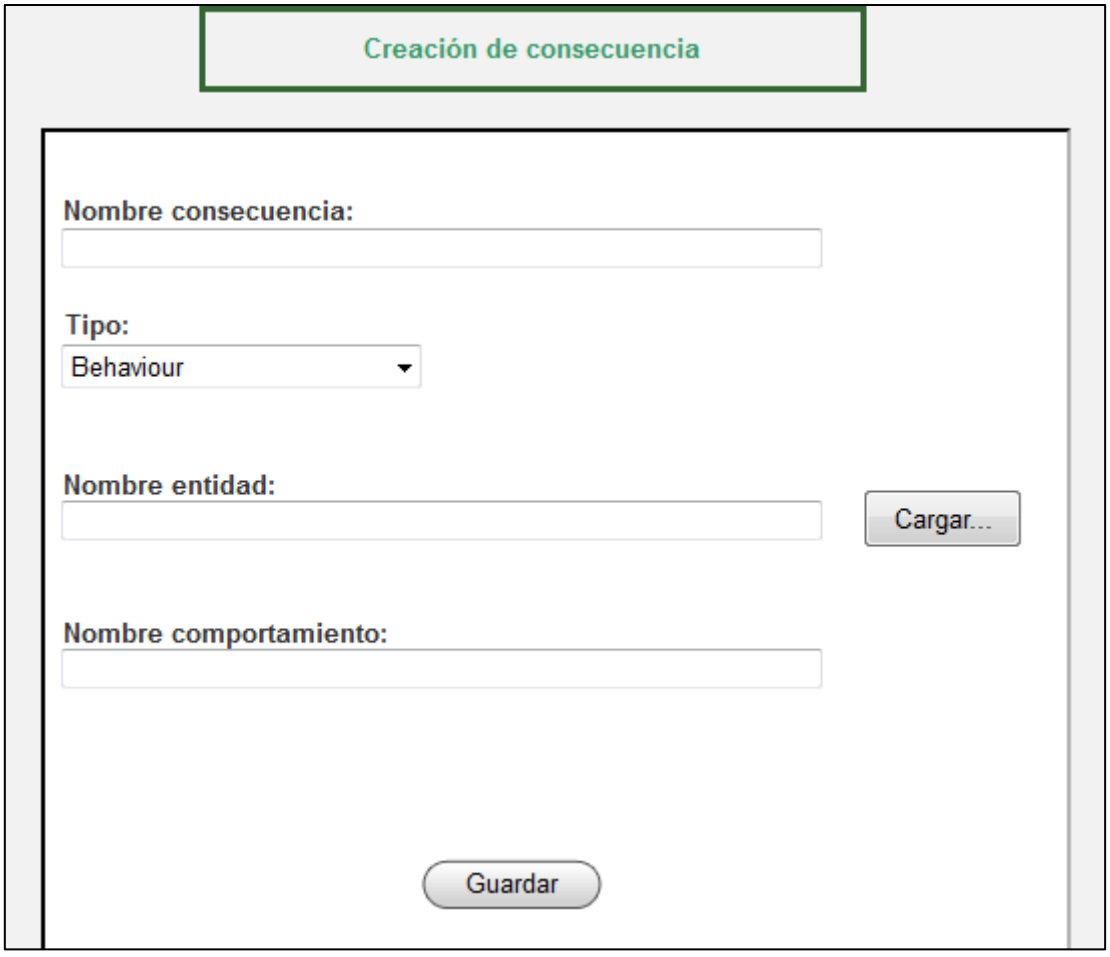


Ilustración 36. Ventana modal de creación de consecuencias

En esta ventana modal es posible la creación de nuevas consecuencias. El formulario correspondiente está formado por estos campos:

- **Nombre consecuencia**. Campo de texto obligatorio en el que se introduce el nombre de la consecuencia.

- **Tipo**. *Combo Box* que permite la selección del tipo de consecuencia.

- **Campos de tipo de consecuencia**. Según el tipo de consecuencia escogido, se mostrarán unos campos u otros.

5.3.2.2.10 Pantalla de definición de objetivos

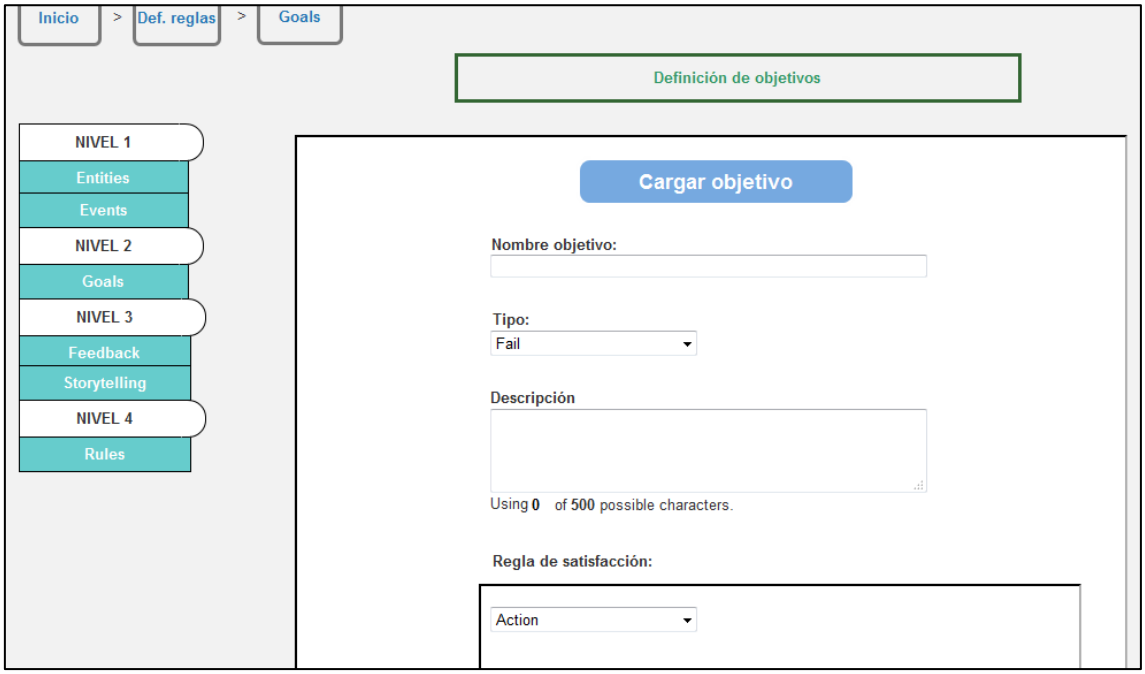


Ilustración 37. Pantalla de objetivos

Esta pantalla, con una distribución similar a la de definición de entidades y eventos, contiene un formulario con los campos necesarios para crear un objetivo, que son:

- **Nombre objetivo**. Campo de texto obligatorio en el que se introduce el nombre del objetivo.

- **Tipo**. *Combo Box* que permite la selección del tipo de objetivo.

- **Descripción**. Área de texto de rellenado opcional en el que se introduce una breve descripción del objetivo.

- **Reglas de satisfacción**. Área en la cual se registra la regla que indica que un objetivo ha sido alcanzado por el jugador. Los campos mostrados se corresponden con las condiciones de evento.

5.3.2.2.11 Pantalla de definición de feedbacks

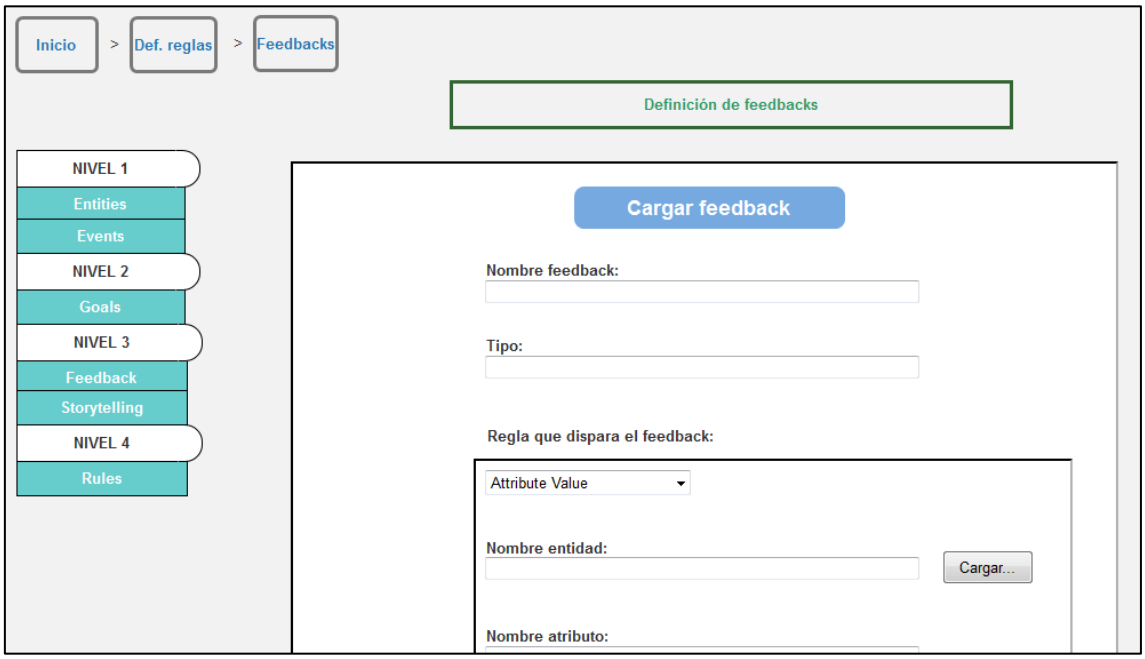


Ilustración 38. Pantalla de feedbacks

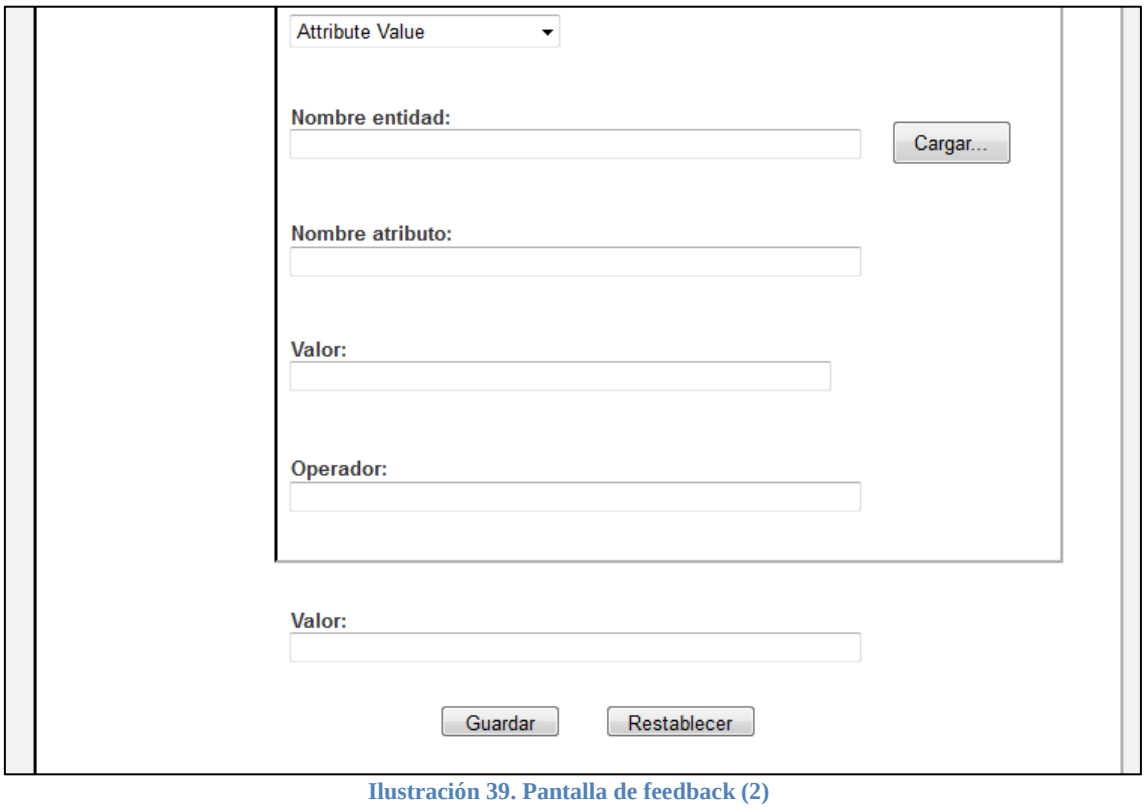


Ilustración 39. Pantalla de feedback (2)

Esta pantalla, con una distribución similar a la de definición de entidades y eventos, contiene un formulario con los campos necesarios para crear un feedback, que son:

- **Nombre feedback**. Campo de texto obligatorio en el que se introduce el nombre del feedback.

- **Tipo**. Campo de texto obligatorio en el que se incluye el tipo de feedback.

- **Reglas de que dispara el feedback**. Área en la cual se registra la regla que indica en qué momento ha de ser mostrado un mensaje de feedback. Los campos mostrados se corresponden con las condiciones de evento.

5.3.2.2.12 Pantalla de definición de storytellings

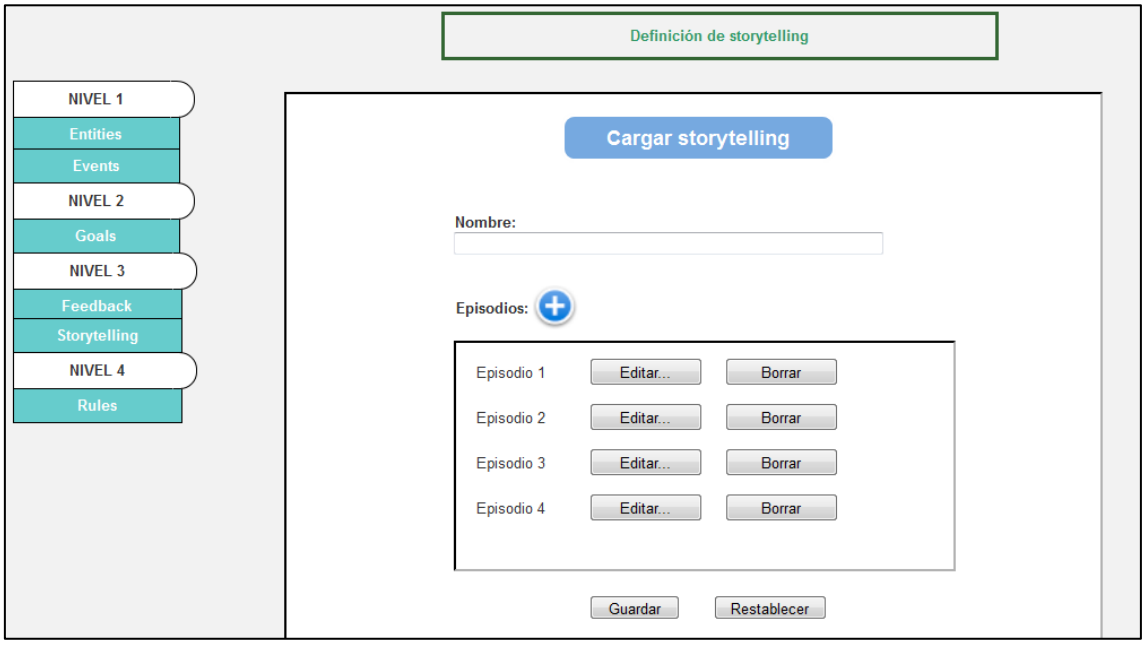


Ilustración 40. Pantalla de storytellings

La ilustración 35 hace referencia a la pantalla que contiene el formulario de creación o edición de storytellings. Siguiendo la estructura de las anteriores páginas de definición, se van a explicar los campos del formulario, que constituyen la principal diferencia:

- **Nombre evento**. Campo de texto obligatorio en el que se introduce el nombre del storytelling.

- **Episodios**. Área en la cual se listan los episodios agregados al storytelling. Se pueden crear episodios nuevos, agregar otros o borrar los que ya estén incluidos.

5.3.2.2.13 Ventana modal de creación de episodios

Ilustración 41. Ventana modal de creación de episodios

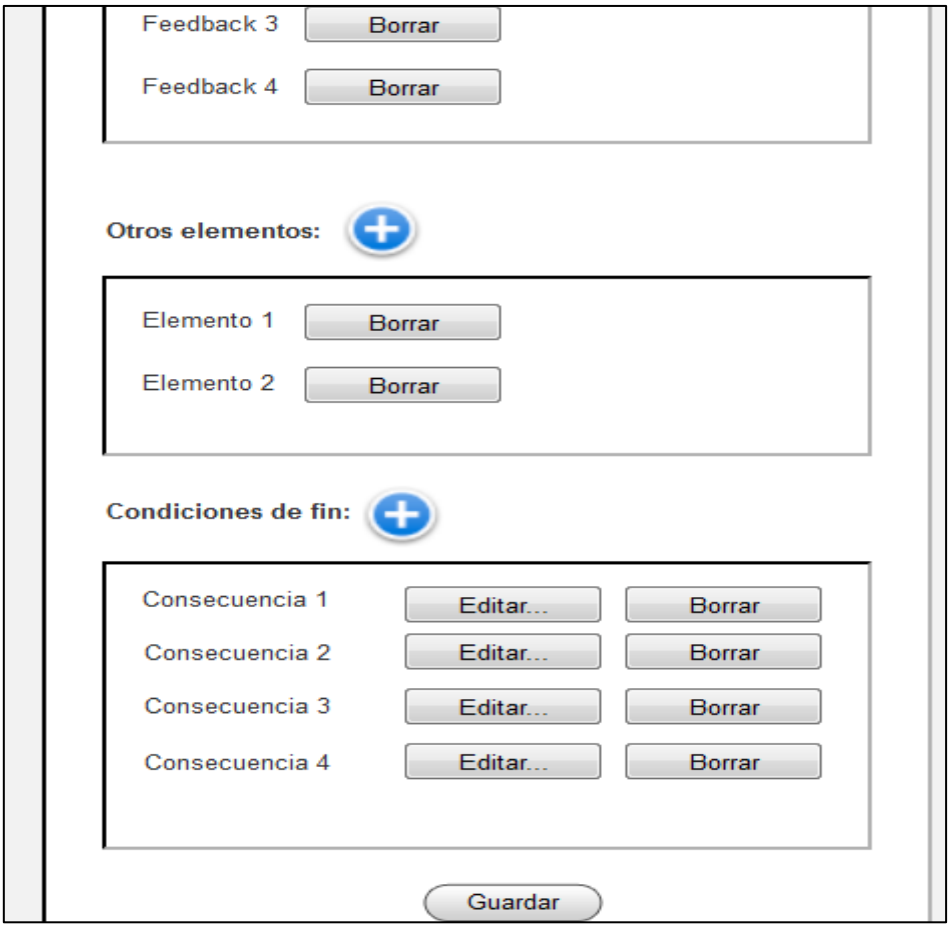


Ilustración 42. Ventana modal de creación de episodios (2)

Esta ventana modal se compone básicamente del formulario que posibilita la creación de nuevos episodios. Está formado por los siguientes campos:

- **ID**. Campo de texto obligatorio en el que se incluye un valor numérico que identifica unívocamente al episodio.
- **Nombre evento**. Campo de texto obligatorio en el que se introduce el nombre del episodio.
- **Entidades**. Área en la cual se listan las entidades agregadas al storytelling. Se pueden agregar otras o borrar las que ya estén incluidas. Existen también las correspondientes áreas para eventos, objetivos, feedbacks y otros elementos.
- **Condiciones de fin**. Área en la que se listan las condiciones de fin añadidas al storytelling. Se pueden crear condiciones de fin nuevas, agregar otras o borrar las que ya estén incluidas.

5.3.2.2.14 Pantalla de definición de conjuntos de reglas y escenarios

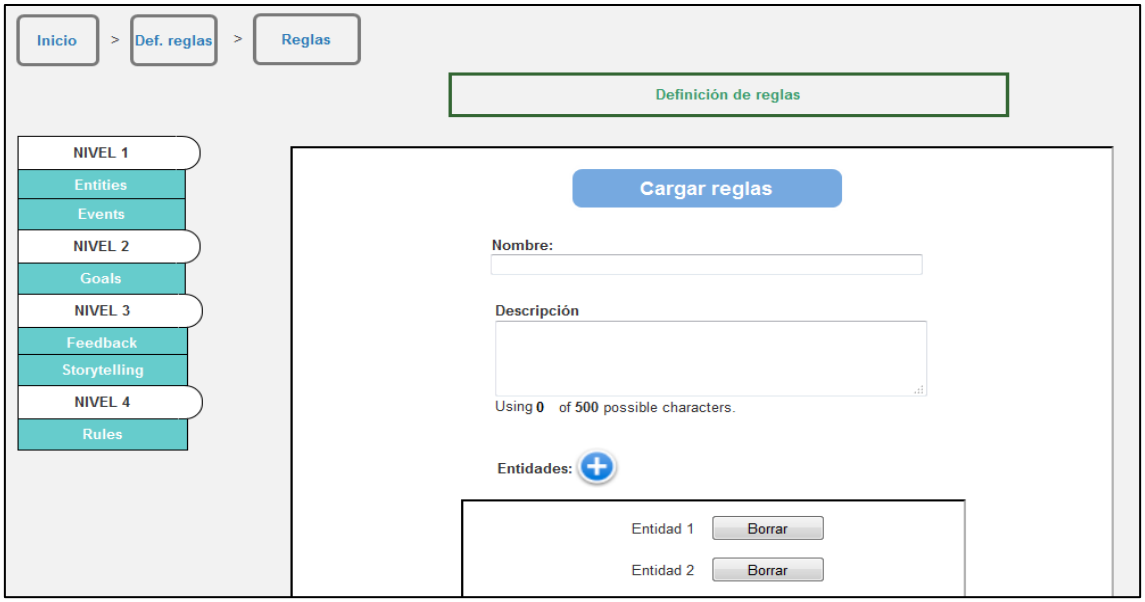


Ilustración 43. Pantalla de reglas y escenarios

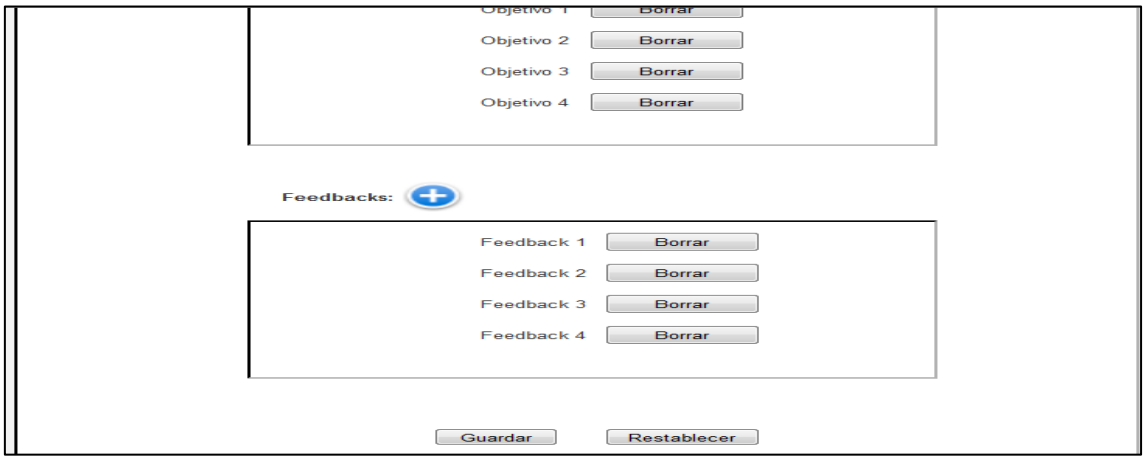


Ilustración 44. Pantalla de reglas y objetivos (2)

La ilustración 39 hace referencia a la pantalla de creación de conjuntos de reglas, aunque es similar a la de escenarios, siendo los elementos que se pueden agregar en cada uno de ellos la diferencia destacable. En el caso de la pantalla de creación de conjuntos de reglas, los elementos que se pueden añadir o borrar una vez agregados son entidades, eventos, objetivos y feedbacks, mientras que en la de escenarios son entidades de escenario, elementos de contexto, servicios e interacciones.

5.3.2.2.15 Pantalla de definición de entidades de escenario

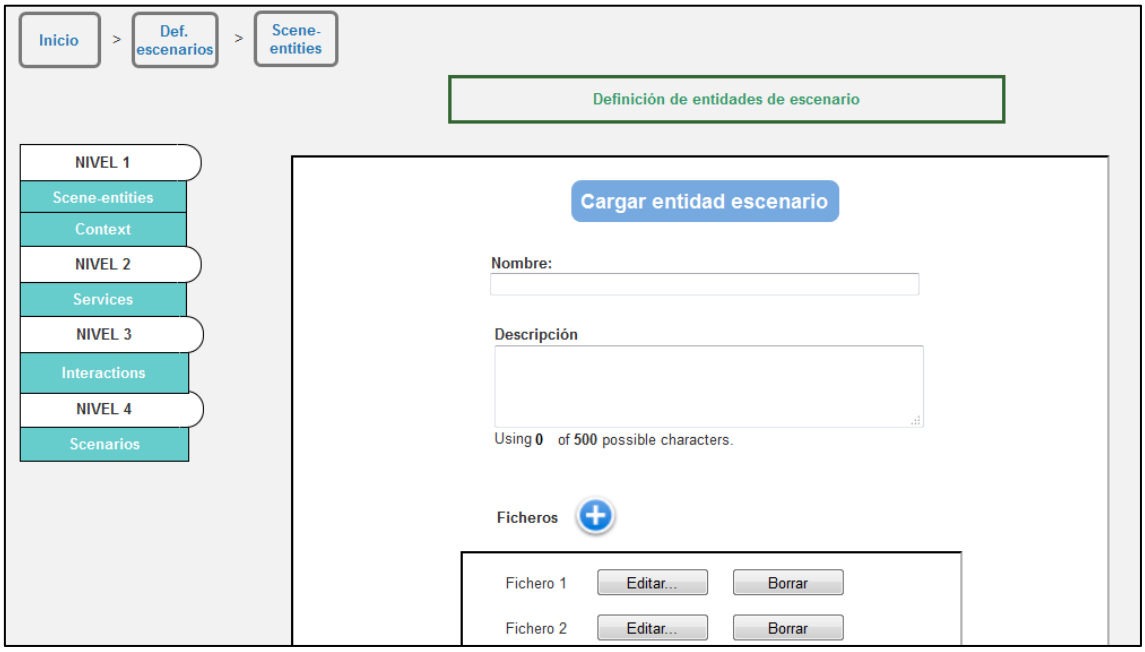


Ilustración 45. Pantalla de entidades de escenario

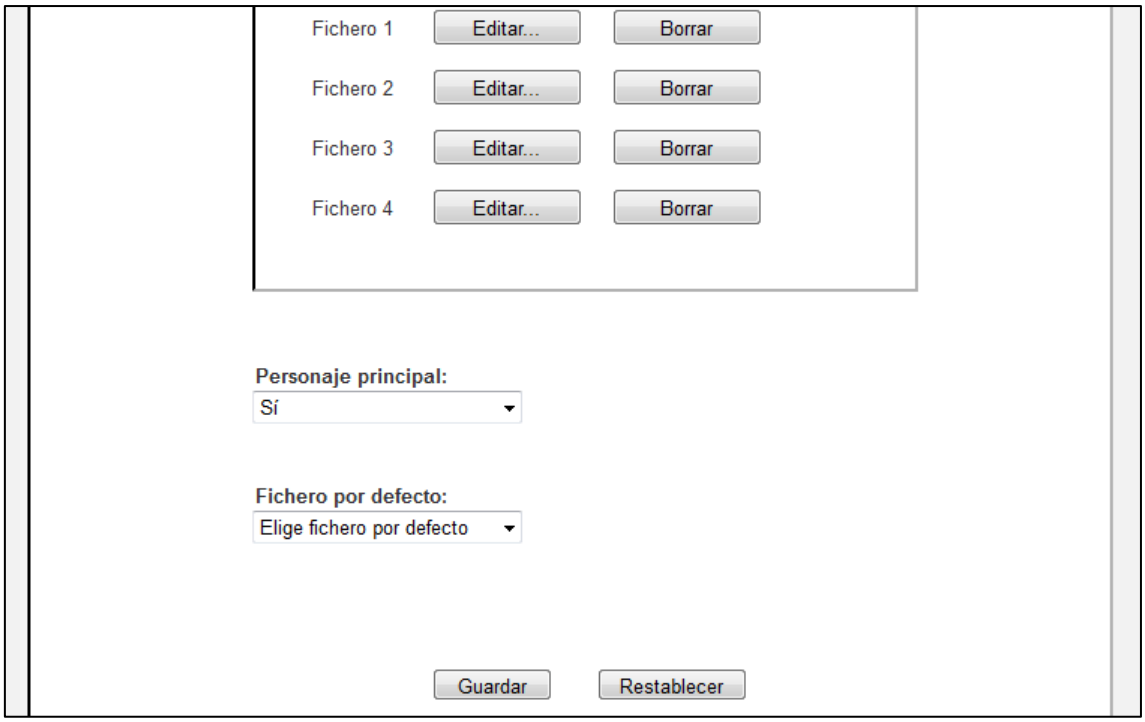


Ilustración 46. Pantalla de entidades de escenario (2)

Las ilustraciones 25 y 26 se corresponden con la pantalla de creación o modificación de entidades de escenario. El mecanismo de "migas de pan" se sigue mostrando debajo de la cabecera del mismo modo que en la pantalla de presentación de secciones. En el lado izquierdo de la página se muestra un menú vertical con todo el recorrido que debería realizarse para obtener un escenario, desde el nivel 1 hasta el nivel 4. Si el usuario pulsa cualquiera de las opciones se le lleva de inmediato a la página de creación del elemento seleccionado. Este menú se adaptará automáticamente a la resolución del dispositivo desde el que se acceda a la aplicación.

En el lado derecho se sitúa el formulario principal de creación o modificación de la entidad. Encima de éste es posible leer unas pequeñas instrucciones que indican cómo rellenar sus campos. En la parte superior del formulario se encuentra un botón que al pulsarlo muestra la ventana modal de carga de una entidad. Los campos del formulario son los siguientes:

- **Nombre**. Campo de texto obligatorio en el que se introduce el nombre de la entidad.
- **Descripción**. Área de texto de rellenado opcional en el que se introduce una breve descripción de la entidad.
- **Ficheros**. Áreas en las que se sitúan los nombres de ficheros de animaciones agregados a la entidad. Se da la posibilidad de eliminarlos de la entidad.
- **Personaje principal**. *Combo box* que indica si la entidad de escenario será manejada por el jugador o no.
- **Fichero por defecto**. *Combo box* que permite la selección de un fichero de animaciones por defecto en caso de que la entidad vaya a ser manejada por el jugador.

En último lugar, se añaden dos botones, uno para enviar la información introducida al controlador correspondiente y tratarla, y otro para limpiar el formulario.

5.3.2.2.16 Ventana modal de creación de nombres de ficheros

Ilustración 47. Ventana modal de creación de nombres de ficheros

La ilustración 42 hace referencia a la ventana modal que muestra el formulario de creación de nombres de fichero. Éste se compone únicamente de un campo obligatorio:

- **Nombre**. Campo de texto obligatorio en el que se introduce el nombre del fichero de animaciones.

5.3.2.2.17 Pantalla de definición de elementos de contexto

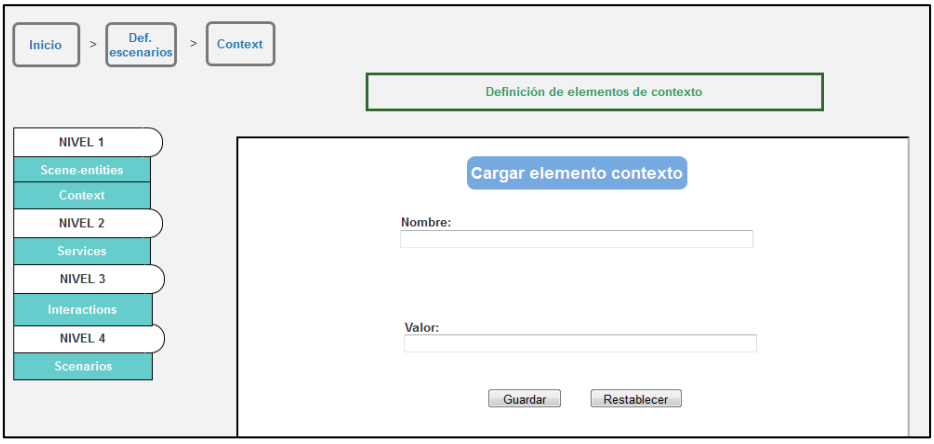


Ilustración 48. Pantalla de elementos de contexto

En esta pantalla el usuario puede crear o editar elementos de contexto. Presenta un aspecto similar al de definición de entidades de escenario, ya que en el lado izquierdo aparece el mismo menú vertical y en el derecho el formulario con los campos a rellenar con los datos del elemento de contexto. La diferencia radica en los propios campos:

- **Nombre**. Campo de texto obligatorio en el que se introduce el nombre del contexto.

- **Valor**. Campo de texto obligatorio en el que se incluye el valor que tendrá el contexto creado.

5.3.2.2.18 Pantalla de definición de servicios

Ilustración 49. Pantalla de servicios

Esta pantalla, con una distribución similar a la de definición de entidades y elementos de contexto, contiene un formulario con los campos necesarios para crear un servicio, que son:

- **Nombre**. Campo de texto obligatorio en el que se introduce el nombre del servicio.
- **Valor**. Campo de texto obligatorio en el que se incluye el valor que tendrá el contexto creado.
- **Posición**. Campo de texto de rellenado opcional en el que se introduce la posición del servicio dentro de la pantalla del videojuego.
- **Tamaño**. Campo de texto de rellenado opcional en el que se incluye el tamaño que ocupa el servicio en la pantalla del videojuego.

5.3.2.2.19 Pantalla de definición de interacciones

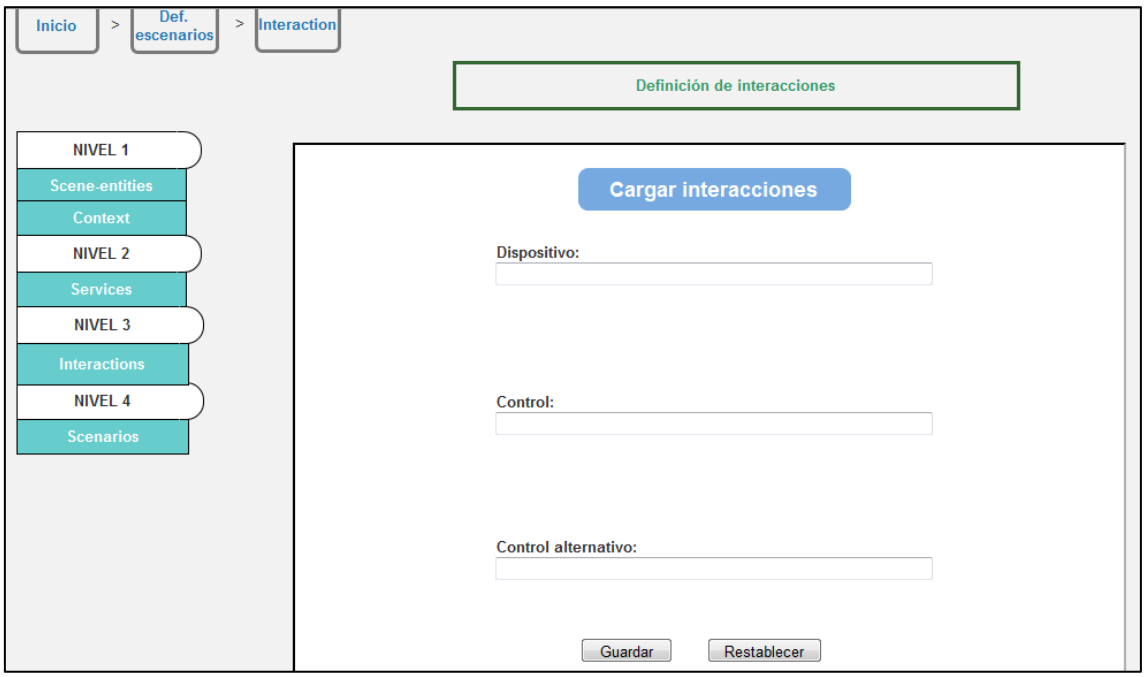


Ilustración 50. Pantalla de interacciones

Esta pantalla, con una distribución similar a la de definición de entidades, elementos de contexto y servicios, contiene un formulario con los campos necesarios para crear una interacción, los cuales son:

- **Dispositivo**. Campo de texto obligatorio en el que se especifica el dispositivo hardware que permitirá la interacción.
- **Control**. Campo de texto obligatorio en el que se indica la manera en que se produce la interacción (pulsando un botón, por ejemplo).

- **Control alternativo**. Campo de texto obligatorio en el que se incluye, como su propio nombre indica, un control alternativo.

5.3.2.2.20 Pantalla de definición de matches

Ilustración 51. Pantalla de matches

La ilustración 46 representa el aspecto que tendría la pantalla de matches, la cual se muestra al pulsar el botón de la pantalla de presentación de la sección correspondiente. Se compone básicamente de un mecanismo de "migas de pan" situado en la esquina superior izquierda bajo la cabecera, de un área de texto que aporta instrucciones acerca de cómo crear un match y generar un XML, y de un formulario. Este formulario presenta la siguiente estructura:

- **Botón de carga de match**. Abre una ventana modal en la que se puede seleccionar un match de los ya creados para consultar su información y generar con ella un documento XML posteriormente.

- **Nombre**. Campo de texto obligatorio en el que se introduce el nombre del match.

- **Ruta XML**. Campo de texto obligatorio en el que se escribe la ruta de destino del documento XML con el diseño del videojuego. Este campo sólo se muestra cuando se carga un match, ya que no tiene sentido incluirlo en el proceso de creación.

- **Botón de carga de conjunto de reglas**. Abre una ventana modal en la que se puede seleccionar el conjunto de reglas para el match.

- **Botón de carga de escenario**. Abre una ventana modal en la que se puede seleccionar el escenario para el match.

- **Tablas**. En cada una de ellas se situarán los datos correspondientes. En la tabla de la izquierda irán los datos del conjunto de reglas, en la del medio los del escenario y en la de la derecha las interacciones creadas.

- **Botones de guardar y restablecer**. Si se pulsa el primero la información se enviará a un controlador para que lleve a cabo el guardado del match en la base de datos. El segundo se encarga de limpiar el formulario. Cuando se carga un match desde base de datos para generar el documento XML, el botón de guardar es sustituido por otro encargado de lanzar la operación de generación.

5.3.2.3 Gestor de operaciones

El último subsistema que queda por describir es el gestor de operaciones. De acuerdo con el patrón MVC, este subsistema será el Controlador. Como tal se encargará de gestionar todos los eventos procedentes de la interfaz de usuario (Vista) y responder a ellos de la manera adecuada.

No obstante, el gestor de operaciones no actúa como un todo, sino que se encuentra dividido en varios componentes que ejecutan funcionalidades específicas. Al observar la ilustración [17], correspondiente al diagrama de componentes, se pueden apreciar cuatro componentes distintos que se describen a continuación:

- **Gestor de reglas**. Este componente se encarga de las labores de creación, edición, borrado, agregación, recuperación y consulta de elementos y subelementos de reglas junto con los conjuntos de reglas propiamente dichos. En otras palabras, abarca la sección correspondiente a la mecánica del videojuego dentro de la herramienta. Proporciona a la Vista una interfaz denominada *iReglas* para permitir la comunicación gestor de reglas-interfaz de usuario.

- **Gestor de escenarios**. Es el componente homónimo del gestor de reglas en la sección asociada a los escenarios del videojuego dentro de la herramienta. A su cargo quedan las funciones de creación, edición, borrado, agregación, recuperación y consulta de elementos y subelementos de escenario junto con los escenarios en sí. Facilita a la Vista una interfaz de nombre *iEscenario* para habilitar la comunicación gestor de escenarios-interfaz de usuario.

- **Gestor de matches**. Se ocupa de los trabajos de creación, borrado, recuperación y consulta de los matches. Proporciona una interfaz llamada *iMatch* para consentir la comunicación entre él mismo y la interfaz de usuario.

- **Gestor de XML**. Su función principal es crear el documento XML a partir de un match cargado desde base de datos. Su interfaz correspondiente, *iXML*, posibilita la comunicación gestor de XML-interfaz de usuario.

Todas las interfaces de los componentes hacen posible la actualización de la interfaz de usuario con los datos tratados. Por otro lado, dichos componentes se comunican con la base de datos a través de una interfaz que proporciona esta última, *iDatos*. De este modo se consigue que los componentes puedan almacenar en la base de datos la información introducida por el usuario y recuperarla si se necesita.

5.3.3 Catálogo de componentes

Para favorecer una mejor comprensión de los subsistemas y componentes que conforman la arquitectura software del sistema, en este apartado se va a desglosar cada uno de ellos de manera tabular. El fin último de todo esto es facilitar la trazabilidad de todos ellos, es decir, si cubren con su propósito todos los requisitos de software funcionales obtenidos en la fase de análisis.

Las tablas que se van a utilizar para situar los componentes van a tener la siguiente estructura:

Identificador	
Nombre	
Tipo	
Propósito	
Función	
Dependencias	
Interfaces	
Subcomponentes	

Tabla 117. Tabla modelo de componentes

Los atributos de la tabla son explicados a continuación:

- **Identificador**. Permite registrar de manera única un componente de acuerdo con una determinada nomenclatura: **C-XX**. **C** hace referencia a un componente. **XX** es un valor numérico que oscila entre 01 y 99.
- **Nombre**. Nombre del componente.
- **Tipo**. Indica la categoría del componente dentro del sistema.
- **Propósito**. Requisitos abarcados en la funcionalidad del componente.
- **Función**. Indica las funcionalidades del componente.
- **Dependencias**. Interfaces requeridas por el componente para llevar a cabo sus tareas.
- **Interfaces**. Interfaces que el componente proporciona para su uso por otros componentes.
- **Subcomponentes**. Especifica los subcomponentes del componente en caso de existir.

Identificador	C-01
Nombre	Interfaz de usuario
Tipo	Subsistema
Propósito	RSF-01, RSF-02, RSF-03, RSF-04, RSF-05, RSF-06, RSF-07, RSF-08, RSF-09, RSF-10, RSF-11, RSF-12, RSF-13, RSF-14, RSF-15, RSF-16, RSF-17, RSF-18, RSF-19, RSF-20, RSF-21, RSF-22, RSF-23, RSF-24, RSF-25, RSF-26, RSF-27, RSF-28, RSF-29, RSF-30, RSF-31, RSF-32, RSF-33, RSF-34, RSF-35
Función	Facilita la comunicación del usuario con el gestor de operaciones a través de eventos y recibe sus respuestas.
Dependencias	iReglas. Posibilita el envío de peticiones de gestión de conjuntos de reglas y sus elementos. iEscenario. Posibilita el envío de peticiones de gestión de escenarios y sus elementos. iMatch. Posibilita el envío de peticiones de gestión de matches. iXML. Posibilita el envío de peticiones de generación de documentos XML.
Interfaces	Ninguna
Subcomponentes	Ninguno

Tabla 118. Componente C-01

Identificador	C-02
Nombre	Gestor de operaciones
Tipo	Subsistema
Propósito	RSF-02, RSF-03, RSF-04, RSF-05, RSF-06, RSF-07, RSF-08, RSF-09, RSF-10, RSF-11, RSF-12, RSF-13, RSF-14, RSF-15, RSF-16, RSF-18, RSF-19, RSF-20, RSF-21, RSF-22, RSF-23, RSF-24, RSF-25, RSF-26, RSF-27, RSF-28, RSF-29, RSF-30, RSF-31, RSF-32, RSF-33, RSF-34, RSF-35
Función	Administra los eventos recibidos de la interfaz de usuario y los responde de la manera adecuada.
Dependencias	Ninguna
Interfaces	Ninguna
Subcomponentes	C-03, C-04, C-05, C-06

Tabla 119. Componente C-02

Identificador	C-03
Nombre	Gestor de reglas
Tipo	Componente
Propósito	RSF-02, RSF-03, RSF-04, RSF-05, RSF-06, RSF-07, RSF-08, RSF-09, RSF-10, RSF-11, RSF-12, RSF-13, RSF-14, RSF-15
Función	Se encarga de las labores de creación, edición, borrado, agregación, recuperación y consulta de elementos y subelementos de reglas junto con los conjuntos de reglas propiamente dichos.
Dependencias	iDatos. Permite la comunicación con la base de datos con el objetivo de modificar o recuperar la información contenida en ella.
Interfaces	iReglas. Posibilita el envío de peticiones de gestión de conjuntos de reglas y sus elementos desde la interfaz de usuario.
Subcomponentes	Ninguno

Tabla 120. Componente C-03

Identificador	C-04
Nombre	Gestor de escenarios
Tipo	Componente
Propósito	RSF-18, RSF-19, RSF-20, RSF-21, RSF-22, RSF-23, RSF-24, RSF-25, RSF-26, RSF-27, RSF-28, RSF-29, RSF-30
Función	Se encarga de las labores de creación, edición, borrado, agregación, recuperación y consulta de elementos y subelementos de escenario junto con los escenarios propiamente dichos.
Dependencias	iDatos. Permite la comunicación con la base de datos con el objetivo de modificar o recuperar la información contenida en ella.
Interfaces	iEscenario. Posibilita el envío de peticiones de gestión de escenarios y sus elementos desde la interfaz de usuario.
Subcomponentes	Ninguno

Tabla 121. Componente C-04

Identificador	C-05
Nombre	Gestor de matches
Tipo	Componente
Propósito	RSF-31, RSF-32, RSF-33, RSF-34
Función	Se ocupa de los trabajos de creación, borrado, recuperación y consulta de los matches.
Dependencias	iDatos. Permite la comunicación con la base de datos con el objetivo de modificar o recuperar la información contenida en ella.
Interfaces	iMatch. Posibilita el envío de peticiones de gestión de matches desde la interfaz de usuario.
Subcomponentes	Ninguno

Tabla 122. Componente C-05

Identificador	C-06
Nombre	Gestor de XML
Tipo	Componente
Propósito	RSF-16, RSF-35
Función	Su función principal es crear el documento XML a partir de un match cargado desde base de datos.
Dependencias	iDatos. Permite la comunicación con la base de datos con el objetivo de modificar o recuperar la información contenida en ella.
Interfaces	iXML. Posibilita el envío de peticiones de generación de documentos XML desde la interfaz de usuario.
Subcomponentes	Ninguno

Tabla 123. Componente C-06

Identificador	C-07
Nombre	Base de datos
Tipo	Subsistema
Propósito	RSF-02, RSF-03, RSF-04, RSF-05, RSF-06, RSF-07, RSF-08, RSF-09, RSF-10, RSF-11, RSF-12, RSF-13, RSF-14, RSF-15, RSF-16, RSF-18, RSF-19, RSF-20, RSF-21, RSF-22, RSF-23, RSF-24, RSF-25, RSF-26, RSF-27, RSF-28, RSF-29, RSF-30, RSF-31, RSF-32, RSF-33, RSF-34, RSF-35
Función	Gestiona la información tratada en la herramienta.
Dependencias	Ninguna
Interfaces	iDatos. Permite la comunicación con la base de datos por parte de los componentes del gestor de operaciones con el objetivo de modificar o recuperar la información contenida en ella.
Subcomponentes	Ninguno

Tabla 124. Componente C-07

5.3.4 Matriz de trazabilidad

En este apartado se presenta la matriz de trazabilidad entre requisitos de software funcionales y componentes para visualizar de una manera rápida qué requisitos son cubiertos por cada uno de los componentes:

REQUISITOS DE SOFTWARE FUNCIONALES	COMPONENTES						
	C-01	C-02	C-03	C-04	C-05	C-06	C-07
RSF-01	X						
RSF-02	X	X	X				X
RSF-03	X	X	X				X
RSF-04	X	X	X				X
RSF-05	X	X	X				X
RSF-06	X	X	X				X
RSF-07	X	X	X				X
RSF-08	X	X	X				X
RSF-09	X	X	X				X
RSF-10	X	X	X				X
RSF-11	X	X	X				X
RSF-12	X	X	X				X
RSF-13	X	X	X				X

RSF-14	X	X	X				X
RSF-15	X	X	X				X
RSF-16	X	X				X	X
RSF-17	X						
RSF-18	X	X		X			X
RSF-19	X	X		X			X
RSF-20	X	X		X			X
RSF-21	X	X		X			X
RSF-22	X	X		X			X
RSF-23	X	X		X			X
RSF-24	X	X		X			X
RSF-25	X	X		X			X
RSF-26	X	X		X			X
RSF-27	X	X		X			X
RSF-28	X	X		X			X
RSF-29	X	X		X			X
RSF-30	X	X		X			X
RSF-31	X	X			X		X
RSF-32	X	X			X		X
RSF-33	X	X			X		X
RSF-34	X	X			X		X
RSF-35	X	X				X	X

Tabla 125. Matriz de trazabilidad entre requisitos de software funcionales y componentes

6. Implementación

En esta sección se comentarán los aspectos más relevantes del proceso de desarrollo de la herramienta. En primer lugar, se presentarán las tecnologías utilizadas en el proceso y se especificará en qué partes del proyecto han sido necesarias. Después se ofrecerá el esquema de archivos del proyecto que ilustra su estructura. Finalmente, se detallarán las partes codificadas más importantes del proyecto y las librerías auxiliares empleadas.

6.1 Tecnologías empleadas

En la implementación de la herramienta han intervenido varias tecnologías que trabajando juntas han conseguido que funcione como se esperaba. A lo largo de esta subsección se explicará con detalle cada una de ellas para obtener una visión global de su utilidad dentro del proceso de desarrollo.

6.1.1 Java Server Pages (JSP)

Las páginas JSP surgen con la idea de facilitar la creación de contenido dinámico a desarrolladores en las aplicaciones sin necesidad de conocer a fondo el lenguaje Java. Estas páginas combinan código HTML con fragmentos de código Java con el objetivo de producir contenido web en el que se mezclan tanto componentes estáticos como dinámicos.

Un servidor de aplicaciones (en el caso de la herramienta, WebSphere) identifica una página JSP por medio de su extensión (.jsp), lo que quiere decir que requiere un tratamiento especial. En este momento, el servidor comprueba si ha sido solicitada con anterioridad. De ser así, significa que la página ya se encuentra en memoria, por lo que sólo es necesario recuperar el código generado. En caso contrario, se notifica al motor de JSP para que genere un servlet para la página. Cuando un cliente solicita una página JSP, se ejecuta en el servidor el código JSP dando como resultado una página HTML que se fusiona con el HTML original, generando una página HTML de respuesta que le será devuelta al cliente.

Tal y como se ha especificado anteriormente, es posible insertar porciones de código Java en la página JSP, denominándose estas porciones *scriptlets*, cuya sintaxis es la siguiente: <% fragmento de código %>.

Esta tecnología ha sido utilizada en la implementación de la interfaz de usuario, que consiste básicamente en un conjunto de páginas JSP que se van mostrando al usuario según sus acciones.

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"
    import="entities.*"
    import="java.util.List"
    import="java.util.ArrayList"
    import="utils.*"
%>
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
    <!-- Para que las páginas se muestren correctamente y el zoom funcione bien en dispositivos móviles -->
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <!--[if lt IE 9]> <script src="http://css3-mediaqueries-js.googlecode.com/svn/trunk/css3-mediaqueries.js"></script> < ![endif]-->
    <!--  Modo de compatibilidad más avanzado en Internet Explorer -->
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="description" content="App Web de diseño de videojuegos">
    <meta name="author" content="Alfredo Alba">

    <title>Scenarios</title>

    <!-- Favicon de la aplicación -->
    <link rel="shortcut icon" href="img/logotipo.png">

    <!-- CSS de Bootstrap -->
    <link href="css/bootstrap.min.css" rel="stylesheet" media="screen">

    <!-- CSS de menús Kickstart (menú vertical) -->
    <link href="css/kickstart/kickstart-menus.css" rel="stylesheet" media="screen">
```

Ilustración 52. Porción de página JSP incluida en la herramienta

En la imagen arriba mostrada se puede apreciar un fragmento de página JSP creada en la herramienta. En la primera línea se aprecia la directiva *page*, que es utilizada para definir y manipular atributos importantes que afectan a la página JSP. En este caso, se indica el lenguaje de los *scriptlets* incluidos en la página, que será Java, la codificación de los caracteres de la página (ISO-8859-1) y los diferentes paquetes Java que serán utilizados. La directiva *page* junto con sus atributos aparecerá en todas las páginas JSP.

Después de dicha directiva se observan elementos comunes en el código HTML, como las etiquetas *meta*, que sirven para añadir información sobre la página o enlaces a imágenes y ficheros CSS. Al igual que con las páginas HTML, las páginas JSP también permiten la inclusión de referencias a archivos JavaScript, las cuales se incluyen al final de la etiqueta *body* acompañadas del atributo *defer* para evitar retardos en el proceso de carga de la página. De este modo, se evita que el código JavaScript contenido en dichos archivos se ejecute antes de que la carga de la página termine.

6.1.2 CSS

Para la definición del estilo de las páginas JSP se han utilizado dos archivos CSS personalizados (*style.css* y *style-responsive.css*) y clases propias del archivo CSS de Bootstrap (*bootstrap.min.css*).

- ***style.css***. En este fichero se ha especificado el estilo general de la herramienta. Se ha elegido una paleta de colores con diferentes tonos de azul que destacan las partes destacables de cada página y variedades de verde para los textos. Los botones principales tienen un color morado para distinguirlos con facilidad. La posición y el tamaño de cada componente también se condensan en este fichero.

- ***style-responsive.css***. Este fichero cumple el cometido de adaptar el estilo especificado anteriormente a la resolución de los dos principales dispositivos en los que se usará la herramienta: ordenadores y *tablets*. El método utilizado para lograr este objetivo consiste en el empleo de consultas de medios (*Media*

Queries), que sirven para aplicar reglas CSS específicas en pantallas. Estas consultas se han indicado explícitamente en el fichero CSS.

```
/* Dispositivos pequeños (tablets, anchura mayor o igual a 768px) y ordenadores medianos (orientación horizontal) */
@media only screen and (min-width : 768px) and (max-width : 1365px) and (orientation : landscape) {

    /*Cabecera de cada página*/
    .jumbotron
    {
        width:100%;
        padding:30px;
        margin-bottom:30px;
        font-size:12px;
        font-weight:200;
        line-height:2.1428571435;
        background-image: -webkit-gradient(
        linear,
        left bottom,
        right bottom,
        color-stop(0.1, #5059FF),
        color-stop(0.22, #3D4DE0),
        color-stop(0.32, #3240FB),
        color-stop(0.98, #121B6B)
        );
        background-image: -o-linear-gradient(right, #5059FF 10%, #3D4DE0 22%, #3240FB 32%, #121B6B 98%);
        background-image: -moz-linear-gradient(right, #5059FF 10%, #3D4DE0 22%, #3240FB 32%, #121B6B 98%);
        background-image: -webkit-linear-gradient(right, #5059FF 10%, #3D4DE0 22%, #3240FB 32%, #121B6B 98%);
        background-image: -ms-linear-gradient(right, #5059FF 10%, #3D4DE0 22%, #3240FB 32%, #121B6B 98%);
        background-image: linear-gradient(to right, #5059FF 10%, #3D4DE0 22%, #3240FB 32%, #121B6B 98%);
        /*background-image: url('../img/fondo_titulo.jpg');*/   /*Fondo azul claro*/
    }

    /*Imagen logotipo*/
    .img-logotipo
    {
```

Ilustración 53. Porción fichero *style-responsive.css*

- ***bootstrap.min.css***. Este fichero proporciona el estilo característico de Bootstrap. Se han utilizado determinadas clases CSS, como *img-responsive*, que adapta una imagen a la resolución del dispositivo en el que se muestre o *row*, que define una fila en la que situar elementos HTML.

6.1.3 Servlets

Un *servlet* es una clase en lenguaje Java usada para ampliar la funcionalidad de los servidores de aplicaciones web y a los que se accede vía modelo de programación *request-response*. Según este modelo, un servlet funciona del siguiente modo:

- Recibe una petición propia del protocolo HTTP (Hypertext Transfer Protocol).
- Traduce dicha petición a una llamada a uno o varios objetos Java que la gestionan.
- Recibe una respuesta de los objetos Java y la traduce a HTTP.
- Devuelve la respuesta a la petición inicial.

En general, las peticiones a *servlets* se responden de una manera rápida al ejecutarse éstos sobre *threads*. De este modo, se propicia el acceso concurrente de los clientes y pueden ser atendidas varias peticiones a la vez.

Los *servlets* se han usado para implementar la funcionalidad de los componentes del gestor de operaciones, ya que ayudan a tratar los datos recibidos de los formularios web

y son los principales responsables de la actualización de la interfaz de usuario al acabar una acción determinada.

6.1.4 Java Persistence API (JPA)

JPA o Java Persistence API es el modelo estándar de Java encargado de automatizar la persistencia de objetos en bases de datos relacionales. Dichos objetos se denominan entidades, las cuales representan tablas en el modelo de datos relacional. Cada instancia de una entidad corresponde a un registro de la tabla que simboliza.

La conexión a la base de datos desde Java se consigue a través del fichero *persistence.xml*, que define el conjunto de entidades que se van a gestionar. Para gestionar dicho conjunto de entidades se utiliza un objeto de la clase *EntityManagerFactory*. A la hora de administrar un determinado tipo de entidad, se utiliza un objeto de la clase *EntityManager* obtenido a partir del *EntityManagerFactory*, que llevará a cabo funciones tan importantes como guardar o borrar las entidades dentro de la base de datos.

Para posibilitar la persistencia de datos en la herramienta, se ha habilitado una conexión con la base de datos y se han añadido las correspondientes entidades a partir de las tablas. A continuación, se han creado automáticamente los *beans* de gestor JPA de cada una de las entidades. Estos *beans* encapsulan y abstraen todo el código de acceso a datos para crear, actualizar, eliminar y consultar información de la base de datos mediante entidades JPA.

A continuación, se profundizará en el código de los *beans* de gestor JPA para estudiar cómo se realizan las consultas y otras operaciones.

6.1.4.1 Consultas

Las consultas a la base de datos serán consultas JPA con nombre (*Named Queries*). Estas consultas son estáticas y tienen la peculiaridad de que en lugar de incorporar literales de forma dinámica en la cadena de consulta se utilizan parámetros predefinidos. Esto constituye una gran ventaja ya que previene posibles ataques de inyección SQL.

Dentro del *bean* de gestor JPA, las consultas con nombre se definen tal y como se observa en la siguiente imagen:

```
protected static final class NamedQueries {

    protected static final String getAllAccionesSinEntidad = "SELECT i FROM Accion i WHERE i.entidad IS NULL";
    protected static final String getAccionesbyEntidad = "SELECT i FROM Accion i WHERE i.entidad = :entidad";
    protected static final String getAccionesConEntidad = "SELECT i FROM Accion i WHERE i.entidad IS NOT NULL";
}
```

Ilustración 54. Formato de las NamedQueries

Para utilizar una consulta con nombre es preciso definir un método de consulta dentro del *bean*, similar al que se puede observar seguidamente.

```
@NamedQueryTarget("getAccionesConEntidad")
public List<Accion> getAccionesConEntidad() {
    EntityManager em = getEntityManager();
    List<Accion> results = null;
    try {
        Query query = em.createQuery(NamedQueries.getAccionesConEntidad);
        results = (List<Accion>) query.getResultList();
    } finally {
        em.close();
    }
    return results;
}
```

Ilustración 55. Método de consulta a BBDD

La anotación *NamedQueryTarget* indica la consulta con nombre que se va a emplear en el método. En este caso concreto, el resultado de la consulta va a ser una lista de entidades de tipo Accion. Para ello, se obtiene la instancia del *EntityManager* asociado a la entidad, se crea la consulta tras acceder a la base de datos y se ejecuta. El resultado de esa consulta es lo que devuelve el método.

6.1.4.2 Inserciones, modificaciones y eliminaciones

Las inserciones, modificaciones y eliminaciones se llevan a cabo dentro del *bean* a través de métodos dedicados: *createEntidadX*, *updateEntidadX* y *deleteEntidadX*, respectivamente.

```
@Action(Action.ACTION_TYPE.CREATE)
public String createAccion(Accion accion) throws Exception {
    EntityManager em = getEntityManager();
    try {
        em.getTransaction().begin();
        em.persist(accion);
        em.getTransaction().commit();
    } catch (Exception ex) {
        try {
            if (em.getTransaction().isActive()) {
                em.getTransaction().rollback();
            }
        } catch (Exception e) {
            ex.printStackTrace();
            throw e;
        }
        throw ex;
    } finally {
        em.close();
    }
    return "";
}
```

Ilustración 56. Método createEntidadX ejemplo

Los tres métodos son similares, exceptuando el método del *EntityManager* que se llama para efectuar la acción correspondiente (*persist* para crear un nuevo registro en la tabla, *merge* para actualizarlo y *delete* para borrarlo). En caso de producirse un error en la transacción de creación, actualización o modificación y permanecer ésta activa se llamará al método *rollback* para deshacer todos los cambios desde la última llamada al método *commit*. De no ser así, se lanzará una excepción.

6.1.5 JavaScript

JavaScript es un lenguaje de programación interpretado y orientado a objetos que se ejecuta en el lado del cliente como parte de un navegador web. La inclusión de código JavaScript en una aplicación web ayuda a mejorar la experiencia con la interfaz de usuario, ya que ejecuta operaciones útiles en el lado del cliente sin tener que recurrir al servidor.

Este lenguaje se ha utilizado como parte de la interfaz de usuario de la herramienta para implementar algunas de sus acciones, como controlar el manejo de las ventanas modales que se muestran o garantizar el dinamismo de los formularios.

```
function mostrarCampos(nombreSelect, valorSelect) {
        /* Si es el Select de Entidad */
        if (nombreSelect == "select_tipo_ent") {
            /* Si se escoge una entidad de tipo PCE, se muestran sus campos correspondientes */
            if (valorSelect == "PC"){
                /* Se muestran los campos de PCE */
                document.getElementById('contenedor_est').style.display = "block";
                document.getElementById('tabla_est').style.display = "block";

                /* Se ocultan los campos de NPCE */
                document.getElementById('contenedor_comp').style.display = "none";
                document.getElementById('tabla_comp').style.display = "none";
            }
            /* Sino, se muestran los de tipo NPCE */
            else{
                /* Se ocultan los campos de PCE */
                document.getElementById('contenedor_est').style.display = "none";
                document.getElementById('tabla_est').style.display = "none";

                /* Se muestran los campos de NPCE */
                document.getElementById('contenedor_comp').style.display = "block";
                document.getElementById('tabla_comp').style.display = "block";
            }
        }
```

Ilustración 57. Porción código JavaScript

6.1.6 jQuery

jQuery es una biblioteca de JavaScript que permite simplificar la manera de trabajar con este lenguaje a la hora de desarrollar aplicaciones del lado del cliente. Básicamente es un compendio de funciones útiles JavaScript con la particularidad de que el código implementado con esta librería es compatible con cualquier navegador web.

Entre otras operaciones, permite una mejor interacción con elementos DOM, gestiona eventos o es capaz de manipular de manera dinámica la hoja de estilos CSS. Su uso en la herramienta se explicará más adelante.

6.2 Esquema de archivos del proyecto

En esta subsección se va a mostrar y detallar el esquema de archivos en que se organiza el proyecto J2EE de la herramienta para entender cómo se organiza el código incluido en él.

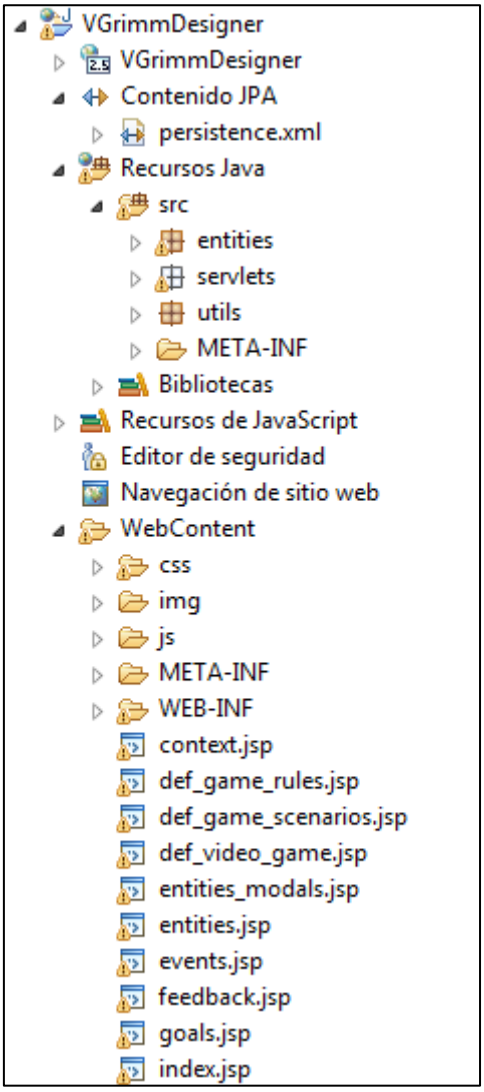


Ilustración 58. Esquema de organización del proyecto

El proyecto *VGrimm Designer* se compone de varias carpetas que contienen diferentes tipos de archivos. Las más importantes son *Recursos Java* y *WebContent*, las cuales se desglosan a continuación:

- ***Recursos Java***. Incluye una subcarpeta de nombre *src*, que a su vez almacena todos los paquetes Java del proyecto. Se distinguen tres paquetes con propósitos distintos.
    - *entities*. Engloba todas las entidades equivalentes a tablas de bases de datos junto a sus correspondientes *beans* de gestor JPA.

- *servlets*. Guarda todos los servlets de la herramienta organizados en subpaquetes según el tipo de elemento del modelo GREM que traten (entidad, evento, etc.).

- *utils*. Comprende un conjunto de clases extra que sirven de apoyo para tratar los datos.

- ***WebContent***. Incluye todas las subcarpetas y archivos requeridos en la parte web de la herramienta, que son:

  - *css*. Almacena todos los ficheros de estilo CSS que se emplean.

  - *img*. Contiene todas las imágenes presentes en la interfaz de usuario.

  - *js*. Engloba todos los archivos JavaScript utilizados.

  - *META-INF*. Presenta en su interior el archivo MANIFEST.MF, que contiene información sobre posibles ficheros *.jar de utilidades incluidos en el proyecto.

  - *WEB-INF*. Contiene los archivos *.*class* de todas las clases Java y servlets creados, el fichero *persistence.xml* y el descriptor de despliegue de la aplicación web (*web.xml*).

  Además de estas subcarpetas, *WebContent* también guarda todos los ficheros JSP de la aplicación.

6.3 Codificación y plugins auxiliares

Tras exponer las tecnologías utilizadas en la implementación y la estructura del proyecto, es momento de comentar los asuntos más importantes en lo que a codificación y uso de plugins se refiere.

6.3.1 Creación y guardado de un conjunto de reglas o de un escenario

La creación y guardado de conjuntos de reglas y escenarios se lleva a cabo en servlets distintos, aunque la funcionalidad es bastante parecida. Dentro de cada uno de estos servlets se localiza un método que en el caso de los conjuntos de reglas se denomina *guardarReglas* y en el de los escenarios es *guardarEscenario*. Ambos métodos son bastante parecidos, por lo que se va a explicar solamente uno de ellos: *guardarReglas*.

```
//Método que ejecuta la acción de guardar un conjunto de reglas
protected void guardarReglas(HttpServletRequest request, HttpServletResponse response, ArrayList<Entidad> listaEntReg,
ArrayList<Evento> listaEvReg, ArrayList<Objetivo> listaObjReg, ArrayList<Feedback> listaFeedReg) throws ServletException, IOException{
    Regla regla = null;
    int idRegla = (int)(10000*Math.random());
    boolean reglasGuardadas = true;
    List<ReglasEntidad> reglas_entidades = managerEntReg.getAllReglasEntidad(); //Se obtienen todas las entradas Reglas_Entidad de BBDD
    List<ReglasEvento> reglas_eventos = managerEvReg.getAllReglasEvento();  //Se obtienen todas las entradas Reglas_Evento de BBDD
    List<ReglasObjetivo> reglas_objetivos = managerObjReg.getAllReglasObjetivo();   //Se obtienen todas las entradas Reglas_Objetivo de BBDD
    List<ReglasFeedback> reglas_feedbacks = managerFeedReg.getAllReglasFeedback();  //Se obtienen todas las entradas Reglas_Feedback de BBDD
    boolean yaGuardadoEnt = false;  //Indica si ya existe una entrada Reglas_Entidad en BBDD
    boolean yaGuardadoEv = false;   //Indica si ya existe una entrada Reglas_Evento en BBDD
    boolean yaGuardadoObj = false;  //Indica si ya existe una entrada Reglas_Objetivo en BBDD
    boolean yaGuardadoFeed = false; //Indica si ya existe una entrada Reglas_Feedback en BBDD

    try {
        //Si las reglas a guardar son unas cargadas previamente
        Object oReg = request.getSession().getAttribute("reglas_cargadas");
        if (oReg != null){
            regla=(Regla)oReg;
            //Se editan las reglas añadiendo los nuevos campos modificados
            regla.setNombre(request.getParameter("nombre_reglas"));
            regla.setDescripcion(request.getParameter("descripcion_reglas"));

            //Se actualiza la entrada de reglas en BBDD con los nuevos datos
            managerReg.updateRegla(regla);
        }
        //Si es un nuevo conjunto de reglas el que se va a guardar se procede a ello
        else{
            //Se crea la instancia de las reglas a guardar
            regla = managerReg.getNewRegla();
            regla.setIdReglas(idRegla);
```

Ilustración 59. Fragmento 1 de creación de reglas

El método recibe como parámetros la *request* y la *response* del servlet para poder devolver la acción a la página que lo ha invocado y una serie de listas con los elementos que formarán parte del conjunto de reglas a crear (entidades, eventos, objetivos y feedbacks). Las primeras líneas de código se corresponden con la inicialización de variables que van a ser usadas a lo largo del métodos, de las cuales son destacables unas listas que almacenarán todos los registros de las tablas auxiliares que contenían el ID de un elemento de reglas y el ID de un conjunto de reglas, los cuales se recuperan a través de unas consultas.

En las siguientes líneas se comprueba si el conjunto de reglas es uno que se ha cargado previamente para consultar sus datos en la página o editar algunos de sus datos. De ser así, se actualizan los campos correspondientes a la entidad JPA del conjunto de reglas con los datos escritos por el usuario en los campos del formulario.

```
            regla.setNombre(request.getParameter("nombre_reglas"));
            regla.setDescripcion(request.getParameter("descripcion_reglas"));

            //Se crea la entrada de las reglas en BBDD
            managerReg.createRegla(regla);
        }

        /*Se guardan las entidades creadas para asociarlas a las reglas. Se creará una entrada en la tabla
        intermedia Reglas_Entidad por cada entidad asociada a unas reglas*/
        if(listaEntReg!=null){
            for(int i=0;i<listaEntReg.size();i++){
                Entidad ent = listaEntReg.get(i);
                yaGuardadoEnt = false;
                if(reglas_entidades!=null){
                    for(int j=0;j<reglas_entidades.size();j++){
                        int idReglasBBDD = reglas_entidades.get(j).getRegla().getIdReglas();
                        int idEntidadBBDD = reglas_entidades.get(j).getEntidad().getIdEntidad();
                        if(idReglasBBDD==regla.getIdReglas() & idEntidadBBDD==ent.getIdEntidad()){
                            yaGuardadoEnt = true;
                            break;
                        }
                    }
                    //Si no existe en BDDD se crea la entrada
                    if(yaGuardadoEnt==false){
                        ReglasEntidad regent = managerEntReg.getNewReglasEntidad();
                        regent.setIdReglas_Entidad((int)(10000*Math.random()));
                        regent.setRegla(regla);
                        regent.setEntidad(ent);
                        managerEntReg.createReglasEntidad(regent);
                    }
                }
                //Si aún no hay nada en BBDD se crea la primera entrada
```

Ilustración 60. Fragmento 2 de creación de reglas

En caso de que se vaya a añadir un nuevo registro en la tabla de reglas, se crea una nueva instancia de la entidad JPA, se añaden los datos escritos por el usuario y se guarda de manera persistente la instancia mencionada para que el registro quede almacenado en la tabla.

Después de estas primeras operaciones, es momento de guardar en la base de datos los elementos que se van a asociar al conjunto de reglas. El procedimiento es similar para todo ellos, así que se va a describir el correspondiente a las entidades:

- En caso de que la lista de entidades no esté vacía se recorre con un bucle *for*.
- Por cada iteración del bucle, se obtiene una entidad y se comprueba si ya existe un registro de la tabla auxiliar correspondiente que asocie dicha entidad con el conjunto de reglas que se va a guardar. En caso de estarlo no se efectuará la acción de guardado.
- Si no existe el registro, se crea uno nuevo. De esta manera, ya quedará establecida la relación entidad-conjunto de reglas, que significa que la entidad forma parte del conjunto de reglas.

```
                }
                //Si aún no hay nada en BBDD se crea la primera entrada
                else{
                    ReglasFeedback regfeed = managerFeedReg.getNewReglasFeedback();
                    regfeed.setIdReglas_Feedback((int)(10000*Math.random()));
                    regfeed.setRegla(regla);
                    regfeed.setFeedback(feed);
                    managerFeedReg.createReglasFeedback(regfeed);
                }
            }
        }
    } catch (Exception e) {
        // TODO Auto-generated catch block
        reglasGuardadas = false;
        e.printStackTrace();
    }

    /*Se vacían las reglas cargadas una vez que están ya guardadas, las entidades, los eventos, los objetivos,
     * los objetivos y el bean de conservación de inputs*/
    request.getSession().setAttribute("reglas_cargadas",null);
    request.getSession().setAttribute("lista_entReglas",null);
    request.getSession().setAttribute("lista_evReglas",null);
    request.getSession().setAttribute("lista_objReglas",null);
    request.getSession().setAttribute("lista_feedReglas",null);
    request.getSession().setAttribute("reglas_nuevas",null);

    //Se indica la modal a abrirse
    if(reglasGuardadas){
        request.setAttribute("nombreModal", "'#modalGuardadoOk'");
    }

    //Se reencamina la petición
    request.getRequestDispatcher("rules.jsp").forward(request, response);
```

Ilustración 61. Fragmento 3 de creación de reglas

Al terminar de guardar en base de datos todos los registros pertinentes, se vacían las variables de sesión que han sido necesarias para almacenar las listas de elementos o el conjunto de reglas. En último lugar, si todo ha salido bien, se le indicará a la interfaz que abra una ventana modal con un mensaje para el usuario indicándole que todo ha ido bien en el momento en que se redireccione a la página de creación de reglas (*rules.jsp*).

6.3.2 Creación y guardado de un match

La creación y guardado de matches se efectúa en un método de un servlet, al igual que ocurría con las reglas y los escenarios. El método concreto es *guardarMatch*, que será detallado seguidamente.

```
//Método que ejecuta la acción de guardar un Match
protected void guardarMatch(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException{
    FinalMatch match = null;
    int idMatch = (int)(10000*Math.random());
    boolean matchGuardado = true;

    /*Se obtienen los contadores que indican el número de entradas que hay de entidades de reglas, de escenarios
    e interacciones*/
    int contadorEntReglas = Integer.parseInt((String)request.getParameter("contadorEntradasEntidadesReglas"));

    //Lista de entradas EntidadEsc-Fichero
    List<EntescFichero> entidadEsc_fichero_bbdd = managerEntescFich.getAllEntescFichero();

    try {
        //Se crea la instancia del match a guardar
        match = managerMatch.getNewFinalMatch();
        match.setIdMatch(idMatch);
        match.setNombre(request.getParameter("nombre_match"));
        match.setEscenario((Escenario)request.getSession().getAttribute("escenario_cargadoMatch"));
        match.setRegla((Regla)request.getSession().getAttribute("reglas_cargadasMatch"));

        //Se crea la entrada del match en BBDD
        managerMatch.createFinalMatch(match);

        //Se guardan los elementos que formarán parte de los MatchEnt que se asociarán al Match

        //Se recorren todas las entradas de entidades de reglas y sus acciones
        for(int i=1;i<=contadorEntReglas;i++){
            //Entidad de reglas
            Entidad entidadReglas = managerEnt.findEntidadByIdEntidad(Integer.parseInt((String)request.getParameter("idEntidadRegla"+i)));
            //Acción asociada
            //Si la acción es "Default action" se deja a NULL el campo correspondiente a la acción
```

Ilustración 62. Fragmento 1 de creación de matches

Este método recibe únicamente como parámetros la *request* y la *response* del servlet para poder devolver la acción a la página que lo ha invocado (*match.jsp*). Las primeras líneas de código se corresponden con la inicialización de variables útiles. Hay que mencionar *contadorEntReglas*, que cuenta el número de entradas de la tabla correspondiente a las entidades de reglas situada en la página JSP, y *entidadEsc_fichero_bbdd*, que recoge de base de datos todos los registros de la tabla auxiliar que establece relaciones entre ficheros y entidades de escenario.

Lo primero que hace el método es crear el registro del match en base de datos. Para ello, se crea una nueva instancia de la entidad JPA del match y añade los campos necesarios, entre ellos el nombre proporcionado por el usuario y los ID´s del conjunto de reglas y del escenario que tomarán parte.

```
String idAccion = (String)request.getParameter("idAccionEntRegla"+i);
Accion accionReglas = null;
if(idAccion.equals("defAction")){
    accionReglas = null;
}
else{
    accionReglas = managerAcc.findAccionByIdAccion(Integer.parseInt(idAccion));
}
//Fila de entidad de escenario
int filaEntidadEsc = Integer.parseInt((String)request.getParameter("id_entidadEscFila"+i));

//A partir de la fila se obtiene la entrada de entidad de escenario correspondiente
//Entidad de escenario
EntidadEscenario entidadEscenario = managerEntEsc.findEntidadEscenarioByIdEntidad_Escenario(Integer.parseInt((String)request.getParamete
//Fichero de escenario
Fichero ficheroEscenario = managerFich.findFicheroByIdFichero(Integer.parseInt(request.getParameter("idFicheroEntEsc"+filaEntidadEsc)));
//Se busca la correspondiente entrada EntradaEsc-Fichero
EntescFichero entEscFich = null;
for(int j=0;j<entidadEsc_fichero_bbdd.size();j++){
    if(entidadEscenario.getIdEntidad_Escenario()==entidadEsc_fichero_bbdd.get(j).getEntidadEscenario().getIdEntidad_Escenario()
    & ficheroEscenario.getIdFichero()==entidadEsc_fichero_bbdd.get(j).getFichero().getIdFichero()){
        entEscFich = entidadEsc_fichero_bbdd.get(j);
        break;
    }
}

//Si la entidad de escenario es Main-Character, significará que tiene interacciones asociadas
Interaccion interaccionEscenario = null;
int filaInteraccionEsc = 0;
if(entidadEscenario.getMainCharacter().equals("si")){
    //Fila de interacción de escenario
    filaInteraccionEsc = Integer.parseInt((String)request.getParameter("id_controlFila"+filaEntidadEsc));
    //Interacción de escenario
```

Ilustración 63. Fragmento 2 de creación de matches

A continuación, se recorren las filas de la tabla de entidades de reglas de la página JSP. En cada iteración se lleva a cabo lo siguiente:

- Se recuperan desde base de datos la entidad y una de sus acciones a partir de sus ID´s. Si la acción resulta ser la que viene por defecto, no se obtiene nada. Tras ello, se recupera el ID de la fila correspondiente de la tabla de entidades de escenario con la que se asociará la fila de la tabla de entidades de reglas.

- A partir de ese ID, se recuperan la entidad de escenario y su correspondiente fichero de animaciones.

- Si por un casual la entidad de escenario será manejada en el videojuego por el jugador, será preciso añadir una interacción. Esta interacción es obtenida a partir del ID de fila de la tabla de interacciones de la página.

```
            interaccionEscenario = managerInt.findInteraccionByIdInteraccion(Integer.parseInt(request.getParameter("idIntEscenario"+filaIn
        }

        //Con los datos obtenidos, se crea la correspondiente fila en la tabla MatchEnt
        MatchEnt matchEnt = managerEntMatch.getNewMatchEnt();
        matchEnt.setIdMatch_Ent((int)(10000*Math.random()));
        matchEnt.setIdColSceneEnt(filaEntidadEsc);
        if(entidadEscenario.getMainCharacter().equals("si")){
            matchEnt.setIdColControl(filaInteraccionEsc);
        }
        else{
            matchEnt.setIdColControl(-1);
        }
        matchEnt.setFinalMatch(match);
        matchEnt.setEntidad(entidadReglas);
        matchEnt.setEntescFichero(entEscFich);
        matchEnt.setAccion(accionReglas);
        matchEnt.setInteraccion(interaccionEscenario);
        managerEntMatch.createMatchEnt(matchEnt);
    }
} catch (Exception e) {
    // TODO Auto-generated catch block
    matchGuardado = false;
    e.printStackTrace();
}

/*Se vacía las reglas cargadas, el escenario cargado y las listas de entidades de reglas y acciones, entidades
 * de escenario y ficheros e interacciones*/
request.getSession().setAttribute("reglas_cargadasMatch",null);
request.getSession().setAttribute("escenario_cargadoMatch",null);
request.getSession().setAttribute("lista_entidadesYAcciones",null);
request.getSession().setAttribute("lista_entidadesYFicheros",null);
request.getSession().setAttribute("lista_intEscenario",null);
```

Ilustración 64. Fragmento 3 de creación de matches

- Una vez que se tienen todos los elementos necesarios, se crea una instancia de entidad JPA *MatchEnt* que representará una fila completa y se hace persistente en la base de datos.

Una vez finalizado el bucle *for*, se liberan todas las variables de sesión utilizadas y se devuelve la acción a la página *match.jsp*.

6.3.3 Generación del documento XML

La generación del documento XML con toda la información de un match se realiza dentro del mismo servlet que la creación de matches; concretamente, en el método *generarXML*.

```
//Método que ejecuta la acción de generar un XML a partir de un Match cargado
protected void generarXML(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException{
    boolean xmlCreado = true;
    DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
    DocumentBuilder builder;
    List<MatchEnt> lista_matchEnt = null;   //Lista de Match_Ent del Match cargado
    List<BeanEntRegEsc> lista_entRegEsc = new ArrayList<BeanEntRegEsc>();   //Lista de entidades de reglas - entidades de escenario que hay en el M
    List<Entidad> lista_entidadReglas = new ArrayList<Entidad>();   //Lista de entidades de reglas (sin repetir)
    Object oMatchEnt = request.getSession().getAttribute("lista_matchEnt"); //Se obtienen los MatchEnt asociados al Match cargado
    if(oMatchEnt!=null){
        lista_matchEnt = (List<MatchEnt>)oMatchEnt;
        //Se obtienen todas las entidades de reglas que hay en el Match (sin repetir)
        for(int i=0;i<lista_matchEnt.size();i++){
            EntescFichero aux = lista_matchEnt.get(i).getEntescFichero();
            EntidadEscenario entEsc = aux.getEntidadEscenario();
            Entidad entReg = lista_matchEnt.get(i).getEntidad();
            BeanEntRegEsc bean = new BeanEntRegEsc();
            bean.setEntidadReglas(entReg);
            bean.setEntidadEscenario(entEsc);
            if(!lista_entidadReglas.contains(entReg)){
                lista_entidadReglas.add(entReg);
                lista_entRegEsc.add(bean);
            }
        }
        try {
            builder = factory.newDocumentBuilder();
            Document document = builder.newDocument();
            Element entities = document.createElement("entities");
            document.appendChild(entities);
            //Se recorren los MatchEnt para generar los elementos del XML
            for(int i=0;i<lista_entRegEsc.size();i++){
                BeanEntRegEsc entIterada = lista_entRegEsc.get(i);
```

Ilustración 65. Fragmento 1 de generación del XML

Este método recibe únicamente como parámetros la *request* y la *response* del servlet para poder devolver la acción a la página que lo ha invocado (*match.jsp*). En las primeras líneas se definen variables útiles a lo largo del método. A continuación, se comprueba que el match contiene filas de elementos. De ser así, tendrá lugar el proceso de generación del documento XML.

En primer lugar se obtienen todas las entidades de reglas que contiene el match sin repetir. Esta repetición de entidades se produce porque por cada una aparece tantas veces dentro de un match como acciones diferentes posea.

Una vez hecho esto, comienza la generación del XML como tal instanciando las variables de tipo *Document*, que representa un documento XML, y *DocumentBuilder*, que ayuda a obtener el XML a partir de la anterior variable.

```
Element entity;
//Si la entidad es Main-Character, se creará la etiqueta <main-character>. En caso contrario, será <entity>
if(entIterada.getEntidadEscenario().getMainCharacter().equals("si")){
    entity = document.createElement("main-character");
    entity.setAttribute("id", entIterada.getEntidadReglas().getNombre());   //El id será el nombre de la Entidad de Reglas
    entity.setAttribute("name", entIterada.getEntidadEscenario().getNombre());  //El name será el nombre de la Entidad de Escenario
}
else{
    entity = document.createElement("entity");
    entity.setAttribute("id", entIterada.getEntidadReglas().getNombre());   //El id será el nombre de la Entidad de Reglas
    entity.setAttribute("name", entIterada.getEntidadEscenario().getNombre());  //El name será el nombre de la Entidad de Escenario
}

//Se obtienen los atributos de la entidad de reglas y se añaden al XML (etiquetas <attribute>)
List<Atributo> lista_atributosEnt = managerAtr.getAtributosbyEntidad(entIterada.getEntidadReglas());
if(lista_atributosEnt!=null){
    for(int j=0;j<lista_atributosEnt.size();j++){
        Element attribute = document.createElement("attribute");
        attribute.setAttribute("type", lista_atributosEnt.get(j).getNombre());
        attribute.setAttribute("value", Integer.toString(lista_atributosEnt.get(j).getValor()));
        entity.appendChild(attribute);
    }
}

//Etiqueta <actions>
Element actions = document.createElement("actions");
//Se añaden todas las acciones de la entidad recorriendo la lista de MatchEnt
for(int j=0;j<lista_matchEnt.size();j++){
    //Si la entidad de reglas del MatchEnt coincide con la entidad que se está iterando se obtienen los datos correspondientes
    if(lista_matchEnt.get(j).getEntidad().getIdEntidad()==entIterada.getEntidadReglas().getIdEntidad()){
        //Etiquetas <action>
        Element action = document.createElement("action");
```

Ilustración 66. Fragmento 2 de generación del XML

```
action.setAttribute("name", lista_matchEnt.get(j).getEntescrichero().getrichero().getNombre()); //name será el nombre del fi
//Si la acción es nula, significará que es la acción por defecto
if(lista_matchEnt.get(j).getAccion()==null){
    action.setAttribute("id", "default");   //id será el nombre de la acción
}
else{
    action.setAttribute("id", lista_matchEnt.get(j).getAccion().getNombre());   //id será el nombre de la acción
}

//Se añade la interacción asociada (etiqueta <control>) a la acción
Element control = document.createElement("control");
//Si la interacción es nula, significará que la acción no tiene interacción
if(lista_matchEnt.get(j).getInteraccion()!=null){
    control.setAttribute(lista_matchEnt.get(j).getInteraccion().getDispositivo(), lista_matchEnt.get(j).getInteraccion().get(
    control.setAttribute("alt", lista_matchEnt.get(j).getInteraccion().getAlt());   //alt
    action.appendChild(control);
}

//Se añaden los requisitos de la acción (etiqueta <requires>)
Accion accionMatch = lista_matchEnt.get(j).getAccion();
if(accionMatch!=null){
    List<Requaccion> lista_reqAccion = managerReq.getRequisitosbyAccion(accionMatch);
    if(lista_reqAccion!=null){
        for(int k=0;k<lista_reqAccion.size();k++){
            Element requires = document.createElement("requires");
            requires.setAttribute("attribute", lista_reqAccion.get(k).getAtributo());   //attribute será el nombre del atrib
            requires.setAttribute("value", Integer.toString(lista_reqAccion.get(k).getValor()));
            action.appendChild(requires);
        }
    }
}
actions.appendChild(action);
}
```

Ilustración 67. Fragmento 3 de generación del XML

```
                entity.appendChild(actions);

                entities.appendChild(entity);
            }

            //Finalmente, se crea el archivo XML en formato fichero
            String ruta = request.getParameter("path_xml"); //Ruta estilo C:/Escritorio/resultado.xml
            Source source = new DOMSource(document);
            Result result = new StreamResult(new java.io.File(ruta));
            Transformer transformer = TransformerFactory.newInstance().newTransformer();
            transformer.transform(source, result);
        } catch (ParserConfigurationException e) {
            // TODO Bloque catch generado automáticamente
            xmlCreado = false;
            e.printStackTrace();
        } catch(Exception e2){
            xmlCreado = false;
            e2.printStackTrace();
        }
    }
    //Se vacían todas las variables de sesión implicadas
    request.getSession().setAttribute("lista_matchEnt", null);
    request.getSession().setAttribute("lista_intEscenarioCarga", null);
    request.getSession().setAttribute("match_cargado", null);

    //Se indica la modal a abrirse
    if(xmlCreado){
        request.setAttribute("nombreModal", "'#modalXMLsuccess'");
    }

    //Se reencamina la petición
    request.getRequestDispatcher("match.jsp").forward(request, response);
```

Ilustración 68. Fragmento 4 de generación del XML

Se inicia entonces un bucle *for* que se ejecuta tantas veces como entidades de reglas existan. Dentro de cada una de esas iteraciones se van generando los elementos del documento XML con ayuda de la variable de tipo *Document*. Tras terminar de crearse todos ellos, una instancia de tipo *Transformer* genera el XML con el formato adecuado y lo sitúa en el directorio especificado por el usuario en el formulario.

Un ejemplo de XML generado sería éste:

```
<?xml version="1.0" encoding="UTF-8"?>
<entities>
    <main-character id="EntidadReglas3" name="EntidadEscenario3">
        <attribute type="Atributo2" value="6644"/>
        <actions>
            <action id="Accion4" name="Fichero4">
                <control alt="d" button="right"/>
                <requires attribute="Atributo1" value="1233"/>
                <requires attribute="Atributo3" value="6777"/>
                <requires attribute="Atributo2" value="5555"/>
            </action>
            <action id="default" name="Fichero4">
                <control alt="d" button="right"/>
            </action>
            <action id="Accion1" name="Fichero3">
                <control alt="a" button="left"/>
                <requires attribute="Atributo1" value="1233"/>
            </action>
            <action id="Accion3" name="Fichero5">
                <control alt="w" button="up"/>
                <requires attribute="Atributo3" value="6777"/>
            </action>
            <action id="Accion2" name="Fichero6">
                <control alt="s" button="down"/>
                <requires attribute="Atributo2" value="5555"/>
            </action>
        </actions>
    </main-character>
    <main-character id="EntidadReglas1" name="EntidadEscenario1">
        <attribute type="Atributo1" value="12345"/>
        <attribute type="Atributo2" value="6644"/>
        <actions>
```

Ilustración 69. Fragmento documento XML generado

6.3.4 Plugins jQuery

Se han empleado dos *plugins* jQuery que han ayudado a llevar a cabo una correcta validación de formularios y a crear tablas con un formato adecuado.

6.3.4.1 jQuery validation plugin

jQuery validation [30] es un *plugin* basado en jQuery que facilita la validación de formularios web. Es un proyecto en constante mejora disponible para su uso libre por parte de cualquier desarrollador de aplicaciones. Con la inclusión en el proyecto de la librería jQuery junto a un fichero *.js denominado *jquery.validate.js* es posible efectuar validaciones bastante completas de una manera sencilla.

Este *plugin* se basa en la utilización de métodos de validación y reglas:

- Un método de validación implementa la lógica para validar un elemento como, por ejemplo, un correo electrónico o el valor de un campo de texto. Se proporcionan algunos métodos estándar de validación, pero es posible crear los que hagan falta sin demasiadas complicaciones.
- Una regla de validación asocia un elemento con un método de validación. Esta asociación se lleva a cabo haciendo uso de los nombres del elemento y del método.

Un ejemplo de validación utilizado en el proyecto sería el siguiente:

```
$('#formulario_goal').validate({
    rules: {
        nombre_goal: {
            required: true,
            maxlength: 30
        },
        select_tipo_goal: {required: true},
        nombre_threshold_goal: {
            required: true,
            digits: true
        }
    },
    messages:{
        nombre_goal: {
            required: '(*) Type a name, please',
            maxlength: 'The name must have less than 30 characters'
        },
        select_tipo_goal: {required: 'Select a type, please'},
        nombre_threshold_goal: {
            required: '(*) Type a value, please',
            digits: 'This field must be numerical'
        }
    }
});
```

Ilustración 70. Ejemplo jQuery validation plugin

Como se puede observar, a través del ID del formulario y haciendo uso del método *validate* del plugin es posible validar todos los campos en pocas líneas de código.

6.3.4.2 Datatables plugin

Datatables [31] es otro *plugin* basado en jQuery que permite dar formato a las tablas HTML mediante una serie de etiquetas con diferentes valores y un estilo CSS que puede ser adaptado al gusto del desarrollador modificando el fichero correspondiente (*jquery.dataTables.css*).

El modo de proceder es sencillo:

- Se incluye el fichero del plugin (*jquery.dataTables*) dentro del proyecto.
- A partir del ID de la tabla a la que se quiera dar un formato concreto, se llama al método *datatable* y se aplican las etiquetas que hagan falta. Todas estas etiquetas vienen explicadas en la documentación de la página web junto a una buena variedad de ejemplos de uso.

Un ejemplo de uso del *plugin* sería este:

```
$('#tablaEntidadesEscenario').dataTable( {
    "scrollY":        "150px",
    "scrollCollapse": true,
    "paging":         false,
    "searching": false,
    "ordering":  false,
    "info": false
} );
```

Ilustración 71. Ejemplo plugin Datatables

La tabla identificada por el ID "tablaEntidadesEscenario" presentará el estilo CSS que se haya definido, una cabecera y un cuerpo y permitirá el *scrolling* vertical al superar los 150px de alto. El resto de etiquetas se ponen a *false* para evitar que se muestren características que vienen por defecto al definir el formato de una tabla con este *plugin*.

7. Pruebas

En esta sección tienen cabida todas las pruebas ejecutadas en el sistema final para verificar el cumplimiento de todos los requisitos. A lo largo del desarrollo se efectuaron pequeños tests de funcionamiento, pero las pruebas que se van a mostrar se corresponden con la batería de pruebas que se llevó a cabo una vez finalizado el sistema en su totalidad.

7.1 Catálogo de pruebas

En vista de una mejor organización de las pruebas efectuadas, cada una de ellas será presentada en formato tabular. Se adjuntarán capturas de pantalla de las pruebas que necesiten una comprobación visual.

El formato de las tablas de pruebas va a ser el siguiente:

Identificador	
Objetivo	
Pasos de ejecución	
Resultado	
Cumplimiento	

Tabla 126. Tabla modelo de pruebas

Los atributos de la tabla son explicados a continuación:

- **Identificador**. Permite registrar de manera única una prueba de acuerdo con una determinada nomenclatura: **P-XX**. **P** hace referencia a una prueba. **XX** es un valor numérico que oscila entre 01 y 99.
- **Objetivo**. Descripción de la finalidad de la prueba.
- **Pasos de ejecución**. Instrucciones guiadas para la ejecución de la prueba.
- **Resultado**. Salida de la prueba.
- **Cumplimiento**. Indica si la prueba ha funcionado o no.

Identificador	P-01
Objetivo	Comprobar el idioma de la aplicación y su compatibilidad con varios navegadores.
Pasos de ejecución	Acceder a la página de inicio de la herramienta de autoría desde varios navegadores.
Resultado	La herramienta está en inglés y se muestra correctamente en Internet Explorer, Mozilla Firefox y Google Chrome.
Cumplimiento	La prueba funciona correctamente.

Tabla 127. Prueba P-01

Ilustración 72. Compatibilidad con Internet Explorer

Ilustración 73. Compatibilidad con Mozilla Firefox

Ilustración 74. Compatibilidad con Google Chrome

Identificador	P-02
Objetivo	Verificar que la herramienta se encuentra protegida frente a ataques XSS.
Pasos de ejecución	1. Acceder a uno de los formularios de la herramienta. 2. Rellenar los campos introduciendo algún tipo de script. 3. Guardar el formulario.
Resultado	La herramienta muestra una página de error en la que se indica que se ha detectado un ataque XSS.
Cumplimiento	La prueba funciona correctamente.

Tabla 128. Prueba P-02

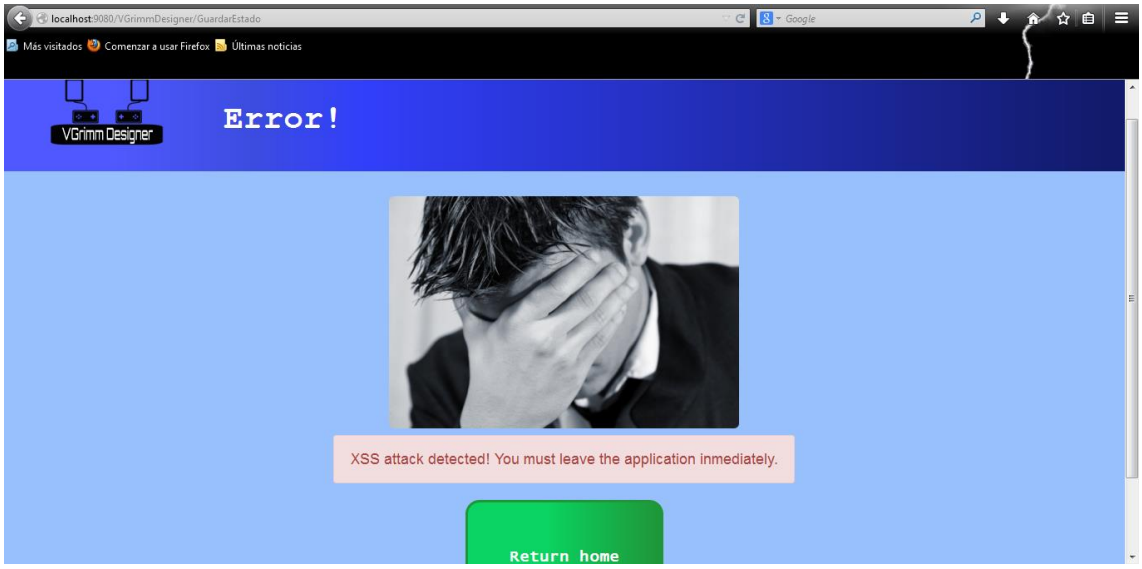


Tabla 129. Ataque XSS detectado

Identificador	P-03
Objetivo	Crear, consultar, editar y borrar un subelemento de reglas como, por ejemplo, un estado.
Pasos de ejecución	1. En la pantalla de entidades de reglas pulsar el botón de crear estados, rellenar los campos necesarios y pulsar el botón de guardado. 2. Pulsar el botón de agregar estados, y hacer click en el botón de editar del estado creado. 3. Cerrar la ventana y volver a pulsar el botón de agregar estados para borrar el estado creado inicialmente.
Resultado	El estado creado inicialmente es borrado del sistema.
Cumplimiento	La prueba funciona correctamente.

Tabla 130. Prueba P-03

Identificador	P-04
Objetivo	Crear, consultar, editar y borrar una entidad de reglas.
Pasos de ejecución	1. En la pantalla de entidades de reglas rellenar los campos de nombre, tipo y descripción del formulario. El último campo no es necesario. 2. Agregar los estados, comportamientos, atributos y acciones oportunos. 3. Pulsar el botón de guardar. 4. Pulsar el botón de cargar entidad, seleccionar la entidad recién creada y pulsar el botón de cargar. 5. Hacer click en el botón de limpiar formulario. 6. Volver a pulsar el botón de cargar entidad y borrar la entidad creada.
Resultado	Se vuelve a mostrar la ventana modal de entidades guardadas y no aparece la entidad borrada.
Cumplimiento	La prueba funciona correctamente.

Tabla 131. Prueba P-04

Identificador	P-05
Objetivo	Crear, consultar, editar y borrar un evento.
Pasos de ejecución	1. En la pantalla de eventos rellenar los campos de nombre y descripción del formulario. El último campo no es necesario. 2. Agregar la condición y las consecuencias del

	evento. **3. Pulsar el botón de guardar.** **4. Pulsar el botón de cargar evento, seleccionar el evento recién creado y pulsar el botón de cargar.** **5. Hacer click en el botón de limpiar formulario.** **6. Volver a pulsar el botón de cargar evento y borrar el evento creado.**
Resultado	**Se vuelve a mostrar la ventana modal de eventos guardados y no aparece el evento borrado.**
Cumplimiento	**La prueba funciona correctamente.**

Tabla 132. Prueba P-05

Identificador	**P-06**
Objetivo	**Crear, consultar, editar y borrar un objetivo.**
Pasos de ejecución	**1. En la pantalla de objetivos rellenar los campos de nombre, tipo, descripción y threshold del formulario. El campo de descripción no es obligatorio.** **2. Agregar un evento al objetivo.** **3. Pulsar el botón de guardar.** **4. Pulsar el botón de cargar objetivo, seleccionar el objetivo recién creado y pulsar el botón de cargar.** **5. Hacer click en el botón de limpiar formulario.** **6. Volver a pulsar el botón de cargar objetivo y borrar el objetivo creado.**
Resultado	**Se vuelve a mostrar la ventana modal de objetivos guardados y no aparece el objetivo borrado.**
Cumplimiento	**La prueba funciona correctamente.**

Tabla 133. Prueba P-06

Identificador	**P-07**
Objetivo	**Crear, consultar, editar y borrar un feedback.**
Pasos de ejecución	**1. En la pantalla de feedbacks rellenar los campos de nombre, tipo y valor del formulario.** **2. Agregar un evento al feedback.** **3. Pulsar el botón de guardar.** **4. Pulsar el botón de cargar feedback, seleccionar el feedback recién creado y pulsar**

	el botón de cargar. 5. Hacer click en el botón de limpiar formulario. 6. Volver a pulsar el botón de cargar feedback y borrar el feedback creado.
Resultado	**Se vuelve a mostrar la ventana modal de feedbacks guardados y no aparece el feedback borrado.**
Cumplimiento	**La prueba funciona correctamente.**

Tabla 134. Prueba P-07

Identificador	**P-08**
Objetivo	**Crear, consultar, editar y borrar un conjunto de reglas.**
Pasos de ejecución	1. En la pantalla de reglas rellenar los campos de nombre y descripción del formulario. El campo de descripción no es obligatorio. 2. Agregar las entidades, eventos, objetivos y feedbacks que se crean necesarios. 3. Pulsar el botón de guardar. 4. Pulsar el botón de cargar reglas, seleccionar el conjunto de reglas recién creado y pulsar el botón de cargar. 5. Hacer click en el botón de limpiar formulario. 6. Volver a pulsar el botón de cargar reglas y borrar el conjunto de reglas creado.
Resultado	**Se vuelve a mostrar la ventana modal de reglas guardadas y no aparece el conjunto de reglas borrado.**
Cumplimiento	**La prueba funciona correctamente.**

Tabla 135. Prueba P-08

Identificador	**P-09**
Objetivo	**Crear, consultar, editar y borrar un subelemento de escenario como, por ejemplo, un nombre de fichero.**
Pasos de ejecución	1. En la pantalla de entidades de escenario pulsar el botón de crear ficheros, rellenar el campo del nombre y pulsar el botón de guardado. El nombre de fichero se muestra en la tabla que hay debajo. 2. Pulsar el botón de editar del nombre de fichero. 3. Cerrar la ventana y borrar el estado creado inicialmente pulsando sobre el botón correspondiente.

Resultado	**El nombre de fichero creado inicialmente es borrado del sistema.**
Cumplimiento	**La prueba funciona correctamente.**

Tabla 136. Prueba P-09

Identificador	**P-10**
Objetivo	**Crear, consultar, editar y borrar una entidad de escenario.**
Pasos de ejecución	**1. En la pantalla de entidades de escenario rellenar los campos de nombre y descripción del formulario. El campo de descripción no es obligatorio.** **2. Agregar los nombres de fichero que se crean necesarios.** **3. Pulsar el botón de guardar.** **4. Pulsar el botón de cargar entidades, seleccionar la entidad recién creada y pulsar el botón de cargar.** **5. Hacer click en el botón de limpiar formulario.** **6. Volver a pulsar el botón de cargar entidad y borrar la entidad creada.**
Resultado	**Se vuelve a mostrar la ventana modal de entidades guardadas y no aparece la entidad borrada.**
Cumplimiento	**La prueba funciona correctamente.**

Tabla 137. Prueba P-10

Identificador	**P-11**
Objetivo	**Crear, consultar, editar y borrar un elemento de contexto.**
Pasos de ejecución	**1. En la pantalla de elementos de contexto rellenar los campos de nombre y valor del formulario.** **2. Pulsar el botón de guardar.** **3. Pulsar el botón de cargar elementos de contexto, seleccionar el elemento recién creado y pulsar el botón de cargar.** **4. Hacer click en el botón de limpiar formulario.** **5. Volver a pulsar el botón de cargar elementos de contexto y borrar el elemento creado.**
Resultado	**Se vuelve a mostrar la ventana modal de elementos de contexto guardados y no aparece el elemento borrado.**

Cumplimiento	**La prueba funciona correctamente.**

Tabla 138. Prueba P-11

Identificador	**P-12**
Objetivo	**Crear, consultar, editar y borrar un servicio.**
Pasos de ejecución	**1. En la pantalla de servicios rellenar los campos de nombre, valor, posición y tamaño del formulario. Los dos últimos campos no son obligatorios. 2. Pulsar el botón de guardar. 3. Pulsar el botón de cargar servicio, seleccionar el servicio recién creado y pulsar el botón de cargar. 4. Hacer click en el botón de limpiar formulario. 5. Volver a pulsar el botón de cargar servicios y borrar el servicio creado.**
Resultado	**Se vuelve a mostrar la ventana modal de servicios guardados y no aparece el servicio borrado.**
Cumplimiento	**La prueba funciona correctamente.**

Tabla 139. Prueba P-12

Identificador	**P-13**
Objetivo	**Crear, consultar, editar y borrar una interacción.**
Pasos de ejecución	**1. En la pantalla de interacciones rellenar los campos de dispositivo, control, control alternativo y tamaño del formulario. 2. Pulsar el botón de guardar. 3. Pulsar el botón de cargar interacción, seleccionar la interacción recién creada y pulsar el botón de cargar. 4. Hacer click en el botón de limpiar formulario. 5. Volver a pulsar el botón de cargar interacciones y borrar la interacción creada.**
Resultado	**Se vuelve a mostrar la ventana modal de interacciones guardadas y no aparece la interacción borrada.**
Cumplimiento	**La prueba funciona correctamente.**

Tabla 140. Prueba P-13

Identificador	P-14
Objetivo	Crear, consultar, editar y borrar un escenario.
Pasos de ejecución	1. En la pantalla de escenarios rellenar los campos de nombre y descripción del formulario. El último campo no es obligatorio. 2. Pulsar el botón de guardar. 3. Pulsar el botón de cargar escenario, seleccionar el escenario recién creado y pulsar el botón de cargar. 4. Hacer click en el botón de limpiar formulario. 5. Volver a pulsar el botón de cargar escenarios y borrar el escenario creado.
Resultado	Se vuelve a mostrar la ventana modal de escenarios guardados y no aparece el escenario borrado.
Cumplimiento	La prueba funciona correctamente.

Tabla 141. Prueba P-14

Identificador	P-15
Objetivo	Crear, consultar, editar y borrar un match.
Pasos de ejecución	1. En la pantalla de matches rellenar el campo de nombre del formulario. 2. Agregar un conjunto de reglas y un escenario. 3. Asociar cada fila de la tabla de entidades de reglas con una de la de entidades de escenario. 4. Asociar las filas de la tabla de entidades de escenario de tipo *main-character* con una fila de la de interacciones. 5. Pulsar el botón de guardar. 6. Pulsar el botón de cargar match, seleccionar el match recién creado y pulsar el botón de cargar. 7. Hacer click en el botón de limpiar formulario. 8. Volver a pulsar el botón de cargar matches y borrar el match creado.
Resultado	Se vuelve a mostrar la ventana modal de matches guardados y no aparece el match borrado.
Cumplimiento	La prueba funciona correctamente.

Tabla 142. Prueba P-15

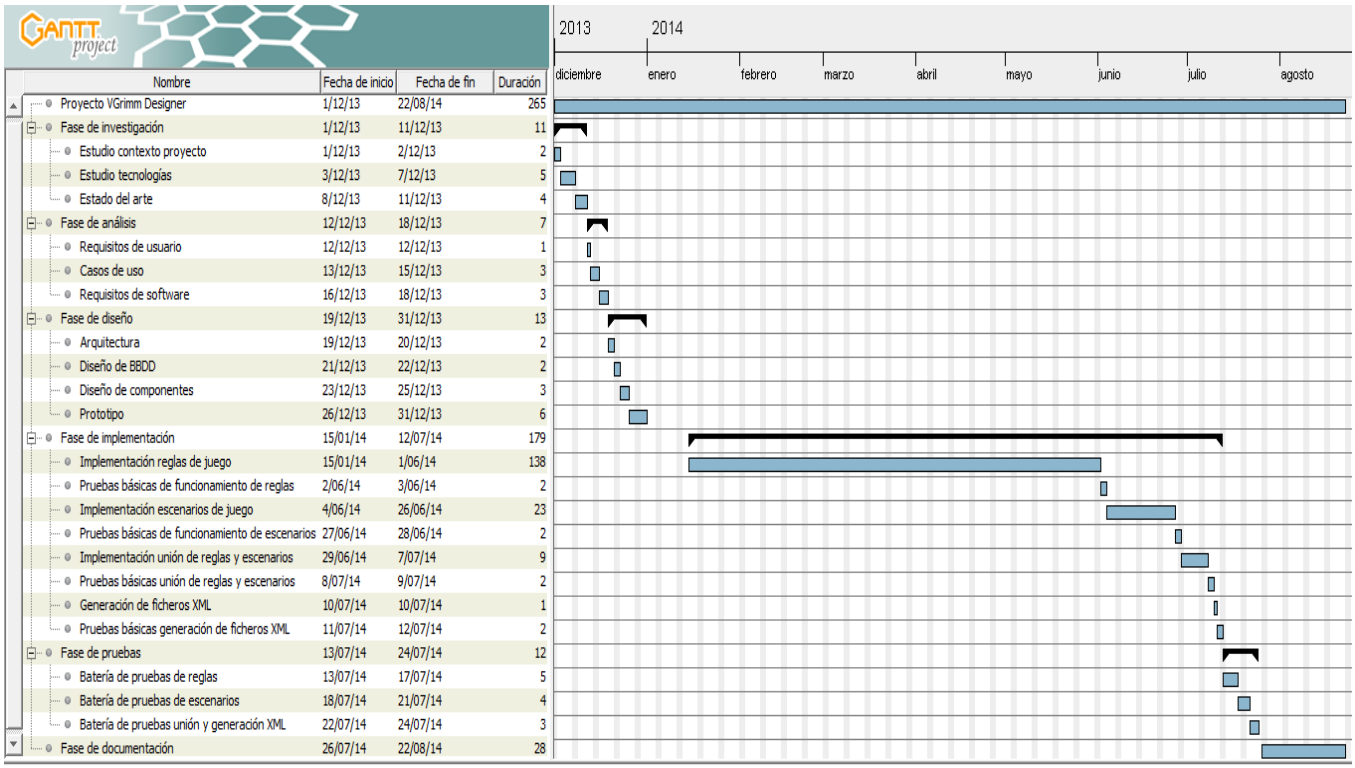
Identificador	P-16
Objetivo	Generar un documento XML a partir de un match creado
Pasos de ejecución	1. En la pantalla de matches pulsar el botón de cargar match. 2. Seleccionar un match de los creados y pulsar el botón de cargar. 3. Rellenar el campo con la ruta destino del documento XML. 4. Revisar los datos y pulsar el botón de generar XML.
Resultado	Se muestra una ventana modal en la que se indica que el documento ha sido generado en la ruta correspondiente.
Cumplimiento	La prueba funciona correctamente.

Tabla 143. Prueba P-16

8. Desviaciones sobre la planificación inicial

En esta sección se evaluarán las desviaciones de tiempo producidas en cada una de las tareas del desarrollo del proyecto con respecto a la planificación inicial que se estableció al comienzo del mismo. Con ello, se identificarán las razones que explican dichas desviaciones y su impacto en la consecución del proyecto.

A continuación, se muestra el diagrama de Gantt correspondiente a la duración final de las tareas una vez acabado el proyecto:



Nombre	Fecha de inicio	Fecha de fin	Duración
Proyecto VGrimm Designer	1/12/13	22/08/14	265
Fase de investigación	1/12/13	11/12/13	11
Estudio contexto proyecto	1/12/13	2/12/13	2
Estudio tecnologías	3/12/13	7/12/13	5
Estado del arte	8/12/13	11/12/13	4
Fase de análisis	12/12/13	18/12/13	7
Requisitos de usuario	12/12/13	12/12/13	1
Casos de uso	13/12/13	15/12/13	3
Requisitos de software	16/12/13	18/12/13	3
Fase de diseño	19/12/13	31/12/13	13
Arquitectura	19/12/13	20/12/13	2
Diseño de BBDD	21/12/13	22/12/13	2
Diseño de componentes	23/12/13	25/12/13	3
Prototipo	26/12/13	31/12/13	6
Fase de implementación	15/01/14	12/07/14	179
Implementación reglas de juego	15/01/14	1/06/14	138
Pruebas básicas de funcionamiento de reglas	2/06/14	3/06/14	2
Implementación escenarios de juego	4/06/14	26/06/14	23
Pruebas básicas de funcionamiento de escenarios	27/06/14	28/06/14	2
Implementación unión de reglas y escenarios	29/06/14	7/07/14	9
Pruebas básicas unión de reglas y escenarios	8/07/14	9/07/14	2
Generación de ficheros XML	10/07/14	10/07/14	1
Pruebas básicas generación de ficheros XML	11/07/14	12/07/14	2
Fase de pruebas	13/07/14	24/07/14	12
Batería de pruebas de reglas	13/07/14	17/07/14	5
Batería de pruebas de escenarios	18/07/14	21/07/14	4
Batería de pruebas unión y generación XML	22/07/14	24/07/14	3
Fase de documentación	26/07/14	22/08/14	28

Ilustración 75. Diagrama de Gantt de la planificación final

En la siguiente tabla se especifican de una manera más clara las diferencias de duración entre las tareas de la planificación inicial y de la planificación final resultante:

Fase	Tarea	Duración estimada (días)	Duración real (días)	Diferencia (días)
Investigación	Estudio contexto proyecto	5 días	2 días	-3 días
	Estudio tecnologías	7 días	5 días	- 2 días
	Estado del arte	4 días	4 días	0 días
Análisis	Requisitos de usuario	2 días	1 día	-1 día

	Casos de uso	**3 días**	**3 días**	**0 días**
	Requisitos de software	**3 días**	**3 días**	**0 días**
Diseño	**Arquitectura**	**4 días**	**2 días**	**-2 días**
	Diseño de BBDD	**3 días**	**2 días**	**-1 día**
	Diseño de componentes	**6 días**	**3 días**	**-3 días**
	Prototipo	**7 días**	**6 días**	**-1 día**
Implementación	**Implementación reglas de juego**	**49 días**	**138 días**	**89 días**
	Pruebas básicas funcionamiento reglas	**2 días**	**2 días**	**0 días**
	Implementación escenarios de juego	**31 días**	**23 días**	**-8 días**
	Pruebas básicas funcionamiento escenarios	**2 días**	**2 días**	**0 día**
	Implementación unión reglas y escenarios	**12 días**	**9 días**	**-3 días**
	Pruebas básicas unión reglas y escenarios	**2 días**	**2 días**	**0 días**
	Generación de ficheros XML	**3 días**	**1 día**	**-2 días**
	Pruebas básicas generación ficheros XML	**2 días**	**2 días**	**0 días**
Pruebas	**Batería de pruebas de reglas**	**5 días**	**5 días**	**0 días**
	Batería de pruebas de escenarios	**5 días**	**4 días**	**-1 día**
	Batería de pruebas unión y generación XML	**3 días**	**3 días**	**0 días**
Documentación	**Documentación de todas las fases del proyecto**	**38 días**	**28 días**	**-10 días**
TOTAL		**197 días**	**265 días**	**52 días**

Tabla 144. Tabla comparativa de planificación

La tabla comparativa anterior muestra la diferencia en días que existe entre la planificación inicial de cada tarea y el tiempo que al final se ha requerido para completarlas. Una diferencia negativa de días en una misma tarea indica un adelanto sobre la planificación inicial, mientras que una positiva conlleva un retraso.

Según los datos de la tabla, el proyecto ha acarreado un total de 265 días desde su inicio hasta su finalización y una desviación de 52 días con respecto a la planificación inicial. Los 16 días de desviación que faltan se corresponden con días no laborables.

El proyecto dio comienzo tal y como estaba planeado el día 1 de diciembre de 2013. Las fases de investigación, análisis y diseño se realizaron antes de lo previsto, lo que conllevaría 13 días extra para la fase de implementación.

No obstante, en la fase de implementación ocurrieron varios sucesos que explican los 52 días de desviación más los 16 no laborables mencionados anteriormente. En primer lugar, existieron problemas a la hora de preparar el entorno de programación escogido, IBM Rational Software Architect. Se realizaron varias consultas al tutor para poner solución a dichos problemas y tras una serie de intentos fallidos se decidió limpiar el disco del equipo y reinstalar el entorno. Al final el entorno funcionó sin más problemas, pero este hecho implicó 15 días de retraso, que no se consideran como tales sino como no laborables.

El siguiente gran retraso se produjo durante la tarea de implementación de reglas de juego. En la planificación inicial se estableció que su duración sería mayor que el resto de tareas de implementación porque en ella también se incluía el proceso de construcción de la interfaz de usuario. Sin embargo, de 49 días se pasó a 138 porque el alumno tuvo que aprender a crear una interfaz con diseño adaptable a distintos dispositivos con ayuda de Bootstrap y CSS. Por otro lado, el uso de las ventanas modales de Bootstrap requirió muchos esfuerzos adicionales de codificación para que su comportamiento dentro de la herramienta fuera intuitivo para el usuario. Finalmente, un error grave que se detectó con más de la mitad de la tarea completada implicó la reestructuración del código ya implementado y la creación de código nuevo.

A principios de mayo el alumno decidió reunirse con el cliente para tratar todos los retrasos producidos y concretar un aplazamiento de entrega del proyecto. Al no tratarse de un proyecto con una fecha de entrega innegociable, el cliente se mostró de acuerdo con la finalización del sistema para el 1 de septiembre. A parte de esto, se decidió dejar de lado la implementación de la sección de storytelling con su correspondiente generación de ficheros de mini-juegos. Desde el principio, el cliente dejó claro con sus requisitos que esta sección tenía una menor prioridad de realización que el resto de las secciones de la herramienta, por lo que su implementación queda entonces abierta como una posibilidad de mejora futura.

Fue también en mayo cuando el alumno tuvo que dejar de trabajar en el proyecto debido a la cercanía de las prácticas y exámenes finales de las asignaturas que cursaba en ese momento. Este periodo de cese se prolongó hasta el 27 de mayo, día en que se retomaron las tareas. A partir de entonces, el resto de tareas y fases se llevaron a cabo en menos días de lo que la planificación inicial estipulaba, por lo que se ganaron unos días de adelanto. Esto se debió a la adaptación completa del alumno a todo el software del que hacía uso.

A pesar de todas las demoras a lo largo del desarrollo del proyecto, el resultado ha sido satisfactorio e incluso mejor de lo esperado, puesto que se aprovechó la prórroga de tiempo hasta la entrega para efectuar mejoras en el sistema.

9. Conclusiones y líneas futuras de trabajo

Esta sección engloba las conclusiones derivadas de la realización de este proyecto así como las posibles líneas futuras de trabajo que podrían surgir a partir de él, ya sea a través de mejoras sobre la herramienta proporcionada o aplicaciones y recursos extra que aumentasen sus prestaciones.

9.1 Conclusiones

En plena Sociedad de la Información no es de extrañar que algunas personas se sientan desbordadas con la gran cantidad de avances tecnológicos que surgen cada día. Los medios de comunicación tal y como se conocían han quedado algo obsoletos tras el acceso al Internet móvil y los dispositivos portátiles. El ocio electrónico se ha convertido en un gran exponente del entretenimiento y era inevitable que su fusión con las tecnologías móviles no se convirtiera en un éxito.

Las posibilidades de los nuevos avances han propiciado que incluso la educación se haya convertido en tema de debate en la actualidad. Los niños de hoy en día se convierten en expertos usuarios de dispositivos electrónicos a una edad cada vez más temprana, por lo que los clásicos métodos de enseñanza ya no despiertan el suficiente interés en ellos. Muchos docentes se han dado cuenta de las posibilidades educacionales que pueden presentar los videojuegos y han comenzado a aplicar sus enseñanzas basándose en ellos cuando son necesarios.

Sin embargo, es complicado acercar los videojuegos a profesores que no disponen de los suficientes conocimientos técnicos para crear los suyos propios o que no confían en la tecnología para impartir la enseñanza. En este hecho reside la importancia de la herramienta de autoría desarrollada en este proyecto, puesto que reduce esa “barrera” que se interpone entre el personal docente y los videojuegos al aplicar un modelo fácil de comprender como es el modelo GREM y un modo de proceder sencillo e intuitivo.

En cuanto a lo personal, la realización de este proyecto ha servido para afianzar la opinión de que los videojuegos no son simplemente un mero entretenimiento, sino un verdadero arte equiparable al cine o la música. Los videojuegos son capaces de emocionar, hacer reír, disfrutar, asustar, motivar e incluso enseñar. Los estereotipos, como en muchos otros aspectos de la vida, son muy dañinos y es necesario dejarlos de lado para conocer las posibilidades que esta industria ofrece.

Numerosos estudios han demostrado que los videojuegos educativos ayudan a mejorar las capacidades motrices y mentales de los niños, por lo que su utilización en las escuelas debería ser cada vez más generalizada. Por esta razón, es de vital importancia que en pleno siglo XXI profesores y padres estén predispuestos a dar una oportunidad a los videojuegos educativos para mejorar el desarrollo de unos niños que van a estar inmersos en una cultura tecnológica.

Respecto a las tecnologías utilizadas para la implementación de la herramienta de autoría, en líneas generales la experiencia ha sido positiva. Java es un lenguaje de programación bastante extendido, fácil de comprender y muy útil, por lo que ha sido el idóneo para la realización de este proyecto. Su combinación con las tecnologías web permite crear aplicaciones bastante dinámicas. Las páginas JSP, en colaboración con los

servlets y JPA, proporcionan muchas posibilidades de desarrollo y facilitan la tediosa tarea de implementar una aplicación desde cero.

Con este proyecto se han aumentado los conocimientos de JavaScript y de librerías jQuery, de uso muy extendido en la actualidad. No obstante, el empleo de un entorno de programación tan complejo como IBM Rational Software Architect puede no haber sido la elección más acertada debido a los inconvenientes derivados de su puesta a punto en el equipo de desarrollo. Pese a todo, es un software muy completo que ha sabido soportar las exigencias del proyecto.

Gracias a este trabajo se han asentado los conocimientos adquiridos a lo largo de la carrera y se han adquirido otros nuevos que serán muy útiles dentro de un ambiente laboral. En muchos momentos, el proyecto ha supuesto un verdadero reto al tener que superar todos los problemas que han ido apareciendo, pero como experiencia ha resultado ser muy beneficiosa.

9.2 Líneas futuras de trabajo

Aunque la herramienta de autoría cumple con su cometido principal es cierto que se podría mejorar con vistas a ofrecer más funcionalidades y ayudarla a ser más completa.

La primera mejora consiste en permitir la creación de ficheros de mini-juegos a partir de storytellings previamente creados, tal y como se estableció en algunos de los requisitos iniciales del proyecto y que no pudieron satisfacerse por una serie de contratiempos. Estos ficheros sirven para conectar diferentes videojuegos e indican los eventos de fin e inicio de cada uno de ellos.

Otra mejora interesante es la inclusión del resto de elementos del modelo GREM para hacer aún más completo el diseño de un videojuego. Estos elementos son Socialización y Debriefing del nivel 3 del submodelo de reglas de juego y Recompensa y Persistencia del nivel 4 de dicho submodelo.

En último lugar, podría aprovecharse la repercusión que tienen las redes sociales y la compartición de archivos en la Nube hoy en día para incluir en la herramienta un servicio que permitiera la interacción entre diferentes docentes dentro de una red de contactos ampliable y otro servicio que facilitara la compartición de los diseños creados por cada uno de ellos.

10. Referencias bibliográficas

[1]. MERA MIRANDA, C. R. (2013). Beneficios del juego en el desarrollo integral de la niñez. *Revista Rayuela* [en línea]. Jun. 2013, nº8. [Consulta: diciembre de 2013]. Disponible en web: <http://revistarayuela.ednica.org.mx/article/beneficios-del-juego-en-el-desarrollo-integral-de-la-ni%C3%B1ez>.

[2]. GONZÁLEZ, D. (2011). Diseño de videojuegos. Da forma a tus sueños. Paracuellos de Jarama, Madrid, España: RA-MA Editorial.

[3]. ZARRAONANDIA, T., DÍAZ, P., AEDO, I., RUÍZ, M. R. (2014). "Designing educational games through a conceptual model based on rules and scenarios", Multimedia Tools and Applications, Ed: Springer US, 1-25.

[4]. eXtensive Markup Language (XML). Accesible en: http://www.w3.org/XML/ [Consulta: diciembre de 2013]

[5]. Unity 3D. Accesible en: http://unity3d.com/ [Consulta: diciembre de 2013]

[6]. SÁNCHEZ IGLESIAS, A.L. (2013). "¿Qué es un Tablet PC?", Accesible en: http://computadoras.about.com/od/tipos-de-pc/a/Que-Es-Un-Tablet-Pc.htm [Consulta: diciembre de 2013]

[7]. Moodle. Accesible en: https://moodle.org/ [Consulta: diciembre de 2013]

[8]. SALVAT, B. G. (2000). La dimensión socioeducativa de los videojuegos. Edutec: Revista electrónica de tecnología educativa, (12), 3.

[9]. BUSTILLO CAVIA, R. (2013). Videojuegos y educación: Un reencuentro necesario. Repositorio Abierto de la Universidad de Cantabria.

[10]. DOMÍNGUEZ, F. I. R. (2004). El poder educativo de los juegos on-line y de los videojuegos, un nuevo reto para la psicopedagogía en la sociedad de la información. Theoria, 13, 97-102.

[11]. Brain Training. Accesible en: http://www.nintendo.es/Juegos/Nintendo-DS/Brain-Training-del-Dr-Kawashima-Cuantos-anios-tiene-tu-cerebro--270627.html [Consulta: diciembre de 2013].

[12]. Civilization. Accesible en: http://www.civilization.com/ [Consulta: diciembre de 2013].

[13]. BENITO, M. (2009). Desafíos pedagógicos de la escuela virtual: las TIC y los nuevos paradigmas educativos. Telos: Cuadernos de comunicación e innovación, (78), 63-77.

[14]. CARRO, R. M., BREDA, A. M., CASTILLO, G., & BAJUELOS, A. L. (2002). Generación de juegos educativos adaptativos. In Actas del III Congreso Internacional de Interacción Persona-Ordenador, Eds. Aedo, I., Cuevas, P., Fernández, C (pp. 164-171).

[15]. MORENO, J., MONTAÑO, E. A., & MONTOYA, L. F. (2012). Creación y monitoreo de video juegos educativos multi-jugador masivos en línea. Conferencias LACLO,3(1).

[16]. TORRENTE VIGIL, F. J., CAÑIZAL ALZOLA, G., & BLANCO AGUADO, Á. D. (2008). Proyecto e-adventure 3D: entorno de autoría para la creación de aventuras 3D educativas en entornos virtuales de enseñanza. E-Prints Complutense (repositorio abierto de la Universidad Complutense de Madrid).

[17]. BEGEGA, S. C., & NISTAL, M. L. (2010). Estudio de plataformas para el desarrollo de juegos educativos de "point and click". Congreso Iberoamericano de Informática Educativa, Volumen 6.

[18]. e-Adventure. Accesible en: http://e-adventure.e-ucm.es/ [Consulta: diciembre de 2013].

[19]. Scratch. Accesible en: http://scratch.mit.edu/ [Consulta: diciembre de 2013].

[20]. Página oficial de jQuery Mobile. Accesible en: http://jquerymobile.com/ [Consulta: diciembre de 2013].

[21]. Página oficial de Bootstrap. Accesible en: http://getbootstrap.com/ [Consulta: diciembre de 2013].

[22]. Página oficial de Foundation. Accesible en: http://foundation.zurb.com/ [Consulta: diciembre de 2013].

[23]. Página oficial de Zend. Accesible en: http://framework.zend.com/ [Consulta: diciembre de 2013].

[24]. Página oficial de CodeIgniter. Accesible en: http://ellislab.com/codeigniter [Consulta: diciembre de 2013].

[25]. Página oficial de HTML Kickstart. Accesible en: http://www.99lime.com/ [Consulta: diciembre de 2013].

[26]. Página oficial de W3C. Accesible en http://www.w3.org/ [Consulta: diciembre de 2013].

[27]. WebSphere Application Server. Accesible en: http://www-03.ibm.com/software/products/es/appserv-was/ [Consulta: diciembre de 2013]

[28]. MySQL. Accesible en: http://www.mysql.com/ [Consulta: diciembre de 2013]

[29]. Axure RP. Accesible en: http://www.axure.com/ [Consulta: diciembre de 2013]

[30]. jQuery validation plugin. Accesible en: http://jqueryvalidation.org/ [Consulta: junio de 2014]

[31]. Datatables plugin. Accesible en: http://www.datatables.net/ [Consulta: junio de 2014]

11. Glosario de términos

Herramienta de autoría: Aplicación informática que facilita la creación y mantenimiento de materiales educativos en formato digital.

CD-ROM: Siglas de *Compact Disc-Read-Only Memory*. Dispositivo de almacenamiento óptico con memoria de sólo lectura con capacidad para guardar hasta 700 MB de datos.

Smartphone: Término procedente del inglés que significa "teléfono inteligente". Teléfono móvil similar a un miniordenador que funciona sobre un sistema operativo y permite, entre otras cosas, el acceso a Internet desde cualquier sitio.

HUD: Siglas de *Heads-Up Display*. Información que se muestra de manera permanente en la pantalla de un videojuego y que recoge datos como salud de un personaje, puntos, etc.

Framework: Término procedente del inglés que significa "marco de trabajo". Entorno tecnológico que proporciona unos módulos concretos para el desarrollo de un determinado tipo de software.

Laptop: Término inglés que hace referencia a un ordenador portátil.

Front-end: Término inglés que se corresponde con la interfaz de usuario y su desarrollo.

Layout: Término inglés que se relaciona con la distribución de los elementos dentro de un diseño.

Match: Unión de un conjunto de reglas de juego y de un escenario.

Directiva: Dentro de JSP, una directiva es una etiqueta que sirve para establecer algún tipo de información acerca de la página, como la importación de una serie de paquetes o una página de error a la que redireccionar.

Thread: En sistemas operativos programa en ejecución que comparte la imagen de memoria con otros threads o hilos. Es la unidad de procesamiento mínima que puede procesar un sistema operativo.

Clase: En programación orientada a objetos, una clase es un conjunto de objetos con el mismo comportamiento.

Objeto: En programación orientada a objetos, un objeto es una instancia de una clase.

Bean: En Java, componente software que tiene la particularidad de ser reutilizable y que facilita la programación.

API: Siglas de *Application Programming Interface*. Conjunto de funciones o métodos que ofrece una biblioteca para ser utilizado por un software.

Transacción: Conjunto de operaciones que deben ser procesadas como una sola unidad.

Fichero JAR: Siglas de *Java ARchives*. Fichero comprimido que almacena otros tipos de archivos Java como los CLASS.

Archivo CLASS: Archivo resultante de la compilación de un archivo JAVA.

Plugin: Aplicación que aporta una función nueva a otra principal.